Apple Assembly Line

$1.80

Volume 6 -- Issue 2 November, 1985

In This Issue...

Little DOS RAM Disk in Language Card 2
Kablit Security System 11
Easier QUIT from ProDOS. 11
Solutions to Adam Levin's Painting Puzzle. 12
Using the Object Vector in S-C Macro Assembler 18
Note on Mainstay MacASM for Macintosh. 19
Commented Listing of ProDOS QUIT Code. 20
Two Ways to Merge Fields in a Byte 28
Comments on O'Ryan's 65C02 mod for Apple II. 32

Programming the 65816

Last month we expected to have ready for this issue a review of David Eyes' new book on programming the 65816 microprocessor. Well the books still haven't arrived, despite the passing of two promised shipping dates, so we're still waiting to see when they will really be available and what they come out like. We are accepting orders (about 20 so far!) and will send out the books and publish a review as soon as they arrive from Prentice-Hall.

quikLoading AppleWorks

For you quikLoader owners who are also using AppleWorks (or for you AppleWorks enthusiasts who want your computer to instantly start up in AppleWorks), Southern California Research Group can now produce a set of quikLoader EPROMs from your configured AppleWorks program disks. The price for the EPROMs and the programming service is $89.50. For more information call SCRG at (805) 529-2082

Little DOS RAM Disk in Language Card.....Bob Sander-Cederlof

For some reason, we have until now avoided this subject. Many versions of RAM disks have been created and published in various magazines. The programs always seemed to me to be rather long and involved for what they really had to do. Recently a friend typed one in from Nibble, prompting me to try my hand.

The so-called "language card" is really the 16K RAM area. In //e and //c computers it is not a separate card at all, just the top 16K of the motherboard RAM. It received the monicker of "language card" because it was first sold as a separate card with the Pascal language system. The RAM in this area is not directly addressable, because the top 16K of Apple's address space is normally allocated to I/O ($C000-CFFF) and ROM ($D000-FFFF).

By flipping a few software-controlled switches the address range from $D000 through $FFFF can be made to point at the 16K RAM instead of ROM. Furthermore, the addresses from $D000 through $DFFF can be pointed at either of two 4K banks. If you have an Apple II or II+ with a 16K RAM card you already know this, of course.

Some programs use the language card under DOS, and some do not. Some which do are Integer BASIC, S-C Macro Assembler, Visicalc, Magicalc, Big Mac, and Merlin. If you are just using Applesoft to run your own programs, the language card is not used.

If the card is otherwise idle, that RAM could be used to simulate a small disk drive. My program sets it up as a 64 sector drive, with 60 sectors available for files. One sector is used for the VTOC, and three sectors are used for the catalog. You can save up to 21 files into the disk, or one file of up to 60 sectors.

One of the first questions I had to answer was where to put the program. Naturally, it ended up at $300. This is almost always my first choice, because it is so easy. If I find some substantial reasons, I try harder and find some other place in RAM for my programs. The ramdisk code could be placed inside DOS itself, on top of the RWTS format code. Another choice might be to use up one page of the language card for the bulk of the code, using only a few lines of code inside RWTS to switch it on and off. I like this idea, but it does deprive me of one sector out of 60. Anyway, for now let's just leave it at $300.

Another choice to be made is how to link into DOS. Many hard disks and other ramdisks do it by placing a JMP or JSR instruction at the beginning of RWTS ($BD00-BD02). This works very well, but it would be nice to be able to use both our ramdisk and any hard disk also. Therefore, I figured out a way to chain my ramdisk together with my Sider hard disk. The method should be compatible with all the ramdisks and hard disks which patch in at $BD00.

S-C Macro Assembler Version 2 0 .....DOS $100, ProDOS $100, both for $120
ProDOS Upgrade Kit for Version 2.0 DOS owners.....$30
Version 2.0 Upgrade Kit for 1 0/1 1/1 2 owners.....$20
Source Code of S-C Macro 2.0 (DOS only).....additional $100
Full Screen Editor for S-C Macro (with complete source code).....$49
S-C Cross Reference Utility.....without source code $20, with source $50
RAK-Ware DISASM.....$30
Source Code for DISASM.....additional $30
S-C Word Processor (with complete source code).....$50
DP18 Source and Object.....$50
Double Precision Floating Point for Applesoft (with source code).....$50
"Bag of Tricks", Worth & Lechner, with diskette.....($39.95) $36 *
MacASM -- Macro Assembler for MacIntosh (Mainstay).....($150.00) $100 *
S-C Documentor (complete commented source code of Applesoft ROMs).....$50
Source Code of //e CX & F8 ROMs on disk.....$15
Cross Assemblers for owners of S-C Macro Assembler ....$32.50 to $50 each
(Available: 6800/1/2. 6301. 6805, 6809, 68000, Z-80. Z-8, 8048,
8051, 8085, 1802/4/5, PDP-11, GI1650/70, others)

AAL Quarterly Disks.....each $15, or any four for $45
Each disk contains the source code from three issues of AAL,
saving you lots of typing and testing.
The quarters are Jan-Mar, Apr-Jun, Jul-Sep, and Oct-Dec.
(All source code is formatted for S-C Macro Assembler. Other assemblers require some effort to convert file type and edit directives.)

Diskettes (with hub rings)..... package of 20 for $32 *
Vinyl disk pages, 6"x8.5", hold two disks each.....10 for $6 *
Diskette Mailing Protectors (hold 1 or 2 disks).....40 cents each
(Cardboard folders designed to fit 6"X9" Envelopes.) or $25 per 100 *
Envelopes for Diskette Mailers..... 6 cents each

65802 Microprocessor (Western Design Center).....($95) $50 *
quikLoader EPROM System (SCRG).....($179) $170 *
PROmGRAMER (SCRG).....($149.50) $140 *
Switch-a-Slot (SCRG).....($190) $175 *
Extend-a-Slot (SCRG).....($35) $32 *
Kablit Security System (Secure-It, Inc.).....($49.95) $45 *

"Programming the 65816", Eyes.....($22.95) $21 *
"Apple //e Reference Manual", Apple Computer.....($24.95) $23 *
"Apple //c Reference Manual", Apple Computer.....($24.95) $23 *
"ProDOS Technical Reference Manual", Apple Computer.....($29.95) $27 *
"Now That You Know Apple Assembly Language...", Gilder.....($19.95) $18 *
"Apple ProDOS: Advanced Features for Programmers", Little..($17.95) $17 *
"Inside the Apple //c", Little.....($19.95) $18 *
"Inside the Apple //e", Little.....($19.95) $18 *
"Apple II+/IIe Troubleshooting & Repair Guide", Brenner.....($19.95) $18 *
"Apple][Circuit Description", Gayler.....($22.95) $21 *
"Understanding the Apple II", Sather.....($22.95) $21 *
"Understanding the Apple //e", Sather.....($24.95) $23 *
"Enhancing Your Apple II, vol. 1", Lancaster.....($15.95) $15 *
"Enhancing Your Apple II, vol. 2", Lancaster.....($17.95) $17 *
"Assembly Cookbook for the Apple II/IIe", Lancaster.....($21.95) $20 *
"Beneath Apple DOS", Worth & Lechner.....($19.95) $18 *
"Beneath Apple ProDOS", Worth & Lechner.....($19.95) $18 *
"6502 Subroutines", Leventhal.....($18.95) $18 *
"Real Time Programming -- Neglected Topics", Foster.....($9.95) $9 *
"Microcomputer Graphics", Myers.....($12.95) $12 *
"Assem Language for Applesoft Programmers", Finley & Myers.($16.95) $16 *
"Assembly Lines -- the Book", Wagner.....($19.95) $18 *
"AppleVisions", Bishop & Grossberger.....($39.95) $36 *

* On these items add $2.00 for the first item and
$.75 for each additional item for US shipping.
Foreign customers inquire for postage needed.
Texas residents please add 6 1/8 % sales tax to all orders.

*** S-C SOFTWARE. P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 ***
*** We accept Master Card, VISA and American Express ***

The program is broken into two parts. The first part installs the ramdisk, and the second part performs the reads and writes. The installer loads and executes at $4000, but of course you could change it to whatever you wish.

I use six page zero locations. These are all locations which are used by regular RWTS, so it is all right for me to use them. I don't even need to save the original data and restore it when I am finished.

Lines 1090-1150 copy the read/write part down to $300-3B4. I actually copy a few extra bytes, but no harm done. I do have to be careful not to write any bytes above $3CF, because $3D0-3FF is already used by DOS and the monitor.

Lines 1160-1230 save the current contents of $BD00-BD02, and place a JMP to my ramdisk code there. Any future calls to RWTS will be vectored to my code down in page 3.

Lines 1250 and 1260 may look ridiculous, if you have not tried programming the language card before. The software-controlled switches ("soft switches") in the Apple are designed so that you have to make two references to address $C083 to turn it on and un-protect it. Two references to $C08B turn on the card also, but with the other 4K bank at $D000.

Lines 1270-1340 store zeroes in every byte from $D000-D3FF. In my scheme, those four pages are equivalent to four sectors (track $11, sectors 0-3). Now that I have mentioned that, why not tell you how I have laid out the whole 16K?

Bank	Addresses	Trk	Sectors
C083	D000-D3FF	$11	0-3
C083	D400-DFFF	$01	4-F
C08B	D000-DFFF	$02	0-F
	E000-EFFF	$03	0-F
	F000-FFFF	$04	0-F

Lines 1350-1420 chain the three catalog sectors together. I have set up track $11 sector 3 as the first catalog sector, sector 2 as the second, and sector 1 as the third and last. This is the same kind of chain DOS makes on a real disk, but shorter.

Lines 1430-1500, together with the two data lines at 1550 and 1560, fill in the non-zero bytes in the VTOC sector. This table driven technique takes somewhat fewer bytes than direct code. I know, because the first time I wrote it the direct way: LDA, STA, LDA, STA, etc. The code as it now is plus the tables takes 45 bytes. The other way it takes 42 bytes just for the STA instructions. If I use LDA #$xx for each of the different values, that is another 16 bytes. So, I saved about 13 bytes. The TBLX line gives the offsets into the $D000 page, and the TBLA line gives the data value which should be stored at each one. I use a 00 offset to indicate the end of the list.

Line 1580 tells the assembler to start assembling code to be executed at $300, but to keep putting the object code bytes in a continuous stream. Since we are writing the code on a target file (see line 1030), the whole program is on one file. RAMDISK.IMAG gets the value $4076, which is what the program counter is BEFORE the .PH directive takes effect. At line 1590 RAMDISK.REAL gets the value $300.

When a program calls RWTS, it is usually through as JSR $B7B5 instruction. The code at $B7B5 disables the interrupts and then does a JSR $BD00. We put our hook at $BD00, so the code jumps to $306, my label LITTLE.RAM.DISK. Lines 1650 and 1660 are the code which normally is executed at $BD00-BD03. They store the IOB address.

Lines 1670-1700 pick up the slot number out of the IOB. This is actually the slot number times 16. If the caller has specified slot 3, he wants to read or write the ramdisk. Any other slot, we need to let regular RWTS do the work. Lines 1710-1750 copy the original contents back to $BD00-BD02. Then I can call RWTS again, and this time it won't come back until it has done its job. Lines 1760-1780 restore Y and A as they were before we got involved, and re-call RWTS. When RWTS is finished, lines 1790-1830 put my hook back into $BD00-BD02. You might wonder if I should be saving and restoring the Y- and A-registers here. I originally did, saving them before line 1790 and restoring them before 1840. Then I realized that the normal contents of Y and A after visiting RWTS are not meaningful. Only the carry status bit is important, as it signifies whether there was an error or not.

If the caller specified slot 3, he wants to talk to our ramdisk. Lines 1860-1900 check to make sure he specified drive 1. If not, we call it an error. I funneled all of the messages through .99, setting the error byte in the IOB to $40. This causes DOS to say there was an I/O error.

I used an EOR #1 rather than CMP #1 at line 189~ so that if the drive was correct, we would also have 0 in the A-register. At some point I need to store 0 into RAMP, and this saves me a LDA #0 instruction. Then line 1910 can set RAMP to 0.

Lines 1930-1970 pick up the sector number the caller specified, and checks it for proper range. It must be from 0 to 15 to be valid. For the time being I save it in a handier location, RAMP+1.

Lines 1980-2020 and 2110-2120 check the track value. I will accept tracks 1-4 and $11, but no others. I have to accept $11, because that is where DOS always expects the VTOC to be, and where the catalog almost always is. The other four tracks could be anything I want, just so they are not $11. Since I am only using 4 sectors of track 11 for VTOC and catalog, I want the others to be usable for files. DOS refuses to allocate any sectors to files in track 11 unless we patch some code in the file manager, so I just put the rest of that bank of ram in another track.

Lines 2040-2060 make sure that if the caller wants track $11, his sector number is not bigger than 3. Lines 2130-2170 make sure that if the caller wants track 1, his sector number is not less than 4. If the track is either $11 or 1, lines 2070-2090 set us up to use the $C083 bank at $D000, with the sector specifying which page in that bank to use.

If the caller wants track 2, 3, or 4 then lines 2250-2310 set up the $C08B side, and compute the page number according to the table given above.

All this may be academic, because we have yet to look at the opcode. We are only implementing read and write, so if the opcode is something else we give an error. Lines 2340-2390 check the opcode, and also set the carry status for read or clear carry for write.

Lines 2400-2420 write enable the ramcard and select the proper $D000 bank. The value in the X-register is either 0 or 8, so we are either addressing $C083 or $C08B twice. We don't really need to write enable it unless the opcode was WRITE, but it doesn't hurt anything.

Lines 2430-2460 clear the error byte in the IOB. I could save two bytes by doing this above, just after line 1910.

Lines 2470-2530 pick up the caller's buffer address and store it in a pointer in page zero. I don't do any range checking on the buffer address, but then neither does RWTS.

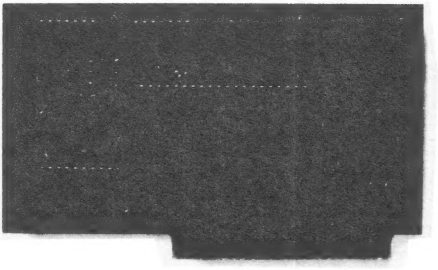
Lines 2540-2550 set Y=0 to start the read or write loop, and then branch to the read loop if carry was set. Lines 2570-2610 comprise the write loop, and lines 2620-2660 the read loop.

Finally, line 2670 turns the language card back off. Then we clear carry status to indicate no errors, and return.

And that is how you make a ramdisk. If you have a bigger RAM card, it probably came with a ramdisk program. But if not, you ought to be able to see how to extend this program to handle larger amounts of memory.

```
                    1000 *SAVE S.LITTLE RAM DISK
                    1010 *-------------------------------
                    1020         .OR $4000
                    1030         .TF B.LITTLE RAM DISK
                    1040 *-------------------------------
3C-                 1050 RAMP    .EQ $3C,3D
3E-                 1060 BUFP    .EQ $3E,3F
48-                 1070 IOB     .EQ $48,49
                    1080 *-------------------------------
                    1090 INSTALL
4000- A0 00         1100         LDY #0        COPY CODE TO PAGE 3
4002- B9 76 40      1110 .0      LDA RAMDISK.IMAG,Y
4005- 99 00 03      1120         STA RAMDISK.REAL,Y
4008- C8            1130         INY
4009- C0 D0         1140         CPY #$D0      NOT PAST $3CF
400B- 90 F5         1150         BCC .0
                    1160 *---INSTALL DOS HOOK------------
400D- A0 02         1170         LDY #2
400F- B9 00 BD      1180 .1      LDA $BD00,Y
4012- 99 00 03      1190         STA OLD.BD00,Y
4015- B9 03 03      1200         LDA NEW.BD00,Y
4018- 99 00 BD      1210         STA $BD00,Y
```

With Z-80 Plus,™ run CP/M®-the largest body of software in existence.



Now, get two computers in one, and all the advantages of both.

Enter the CP/M world with the new Z-80 Plus card from Applied Engineering, and introduce your Apple IIe® or II + ® to the thousands of CP/M programs. Only the Z-80 Plus comes standard with the new 4.0 software, the most advanced system ever for running CP/M programs.

The new 4.0 boasts advanced features like built-in disk emulation for popular memory expansion boards, boosting both system speed and storage capacity. And menu-driven utilities that let you get to work faster. The Z-80 Plus also lets you run older CP/M programs — all the way down to Version 1.6 (2.2 is the most popular).

The Z-80 Plus is the only card on the market capable of accessing more than 64K in an Apple IIe. If you have an extended 80-column card, all 128K is usable, and if you have RamWorks, up to 1088K is available.

Each Z-80 Plus comes with our CP/M Ram Drive software, enabling IIe owners to use an extended 80-column card or a RamWorks card as a high-speed Ram disk which runs CP/M software up to *twenty times faster*. So packages like WordStar and dBASE II run at blinding speed.

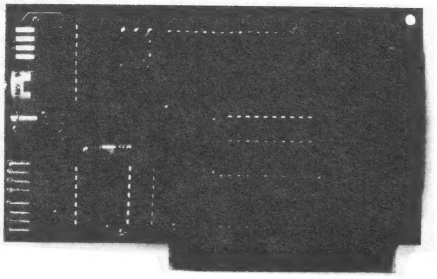
Simply plug the Z-80 Plus into any slot in your Apple. You'll get the benefits of two computers in one — all at an unbelievably low price (only $139!).

• Fully compatible with ALL CP/M software • Fully compatible with most hard disks, including Corvus and the Sider • Fully compatible with Microsoft disks (no pre-boot required) • Specifically designed for high speed operation in the Apple IIe (runs just as fast in the Apple II + and Franklin) • Runs WordStar, dBASE II, Turbo Pascal, Fortran-80, Peachtree and ALL other CP/M software with *no pre-boot* • Semi-custom I.C. and low parts count allows Z-80 Plus to fly through CP/M programs with extremely low power consumption (we use the Z-80B) • Does EVERYTHING other Z-80 boards do, *plus* Z-80 interrupts • Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

Timemaster H.O.,™ the only clock that displays time and date on AppleWorks'™ screens and files.



Now, get all the features of all the competition combined!

It's the smart way to put the time and date on your Apple II + ® or IIe®. Because only the Timemaster H.O. packs *ALL* the features of all the competition *combined*, including leap year, year (not just in PRO-DOS), month, date, day of week, hours, minutes, seconds and milliseconds. It's totally PRO-DOS, DOS 3.3, PASCAL and CP/M compatible. And of course, it works better than any other clock with AppleWorks.

If you're using or writing software for other clock cards, you're still covered. Because the H.O. will *automatically* emulate them. And the Timemaster H.O. adds 14 new commands to BASIC. The H.O. even comes complete with two disks full of sample programs, including a computerized appointment book, a DOS dating program, interrupt programs, and over 30 programs that others charge extra for — or don't even offer.

As a low-cost option, you can add true BSR remote control to the H.O., giving you remote control of up to 16 lights and appliances in your home or office.

• Fully PRO-DOS and DOS 3.3, CP/M and PASCAL compatible • Time in hours, minutes, seconds and milliseconds (the ONLY PRO-DOS compatible card with millisecond capability); date with year, month, day of week and leap year • 24-Hour military format or 12-hour AM/PM format • Eight software controlled interrupts so you can run two programs at the same time (many examples included) • Allows AppleWorks to time and date stamp all data automatically • The only clock card that displays time and date on the AppleWorks screen • Five year warranty

Clock price **$129.00**
BSR option (may be added later) ..... **$ 49.00**

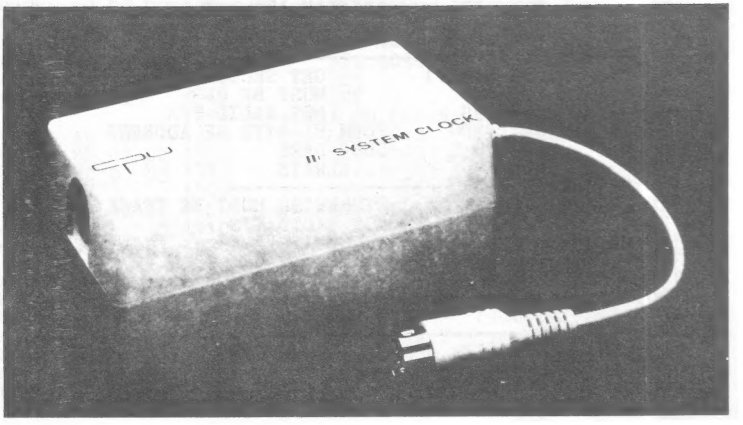
Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

```
401B- 88          1220         DEY
401C- 10 F1       1230         BPL .1
                  1240 *---INIT VTOC & CATALOG----------
401E- AD 83 C0    1250         LDA $C083
4021- AD 83 C0    1260         LDA $C083
4024- C8          1270         INY              Y=0
4025- A8          1280         TAY
4026- 99 00 D0    1290 .2      STA $D000,Y  CLEAR VTOC
4029- 99 00 D1    1300         STA $D100,Y  CLEAR THREE CATALOG PAGES
402C- 99 00 D2    1310         STA $D200,Y  ...ROOM FOR 21 FILES
402F- 99 00 D3    1320         STA $D300,Y
4032- C8          1330         INY
4033- D0 F1       1340         BNE .2
                  1350 *---CATALOG CHAIN----------------
4035- A9 11       1360         LDA #$11     SIMULATED TRACK 11
4037- 8D 01 D2    1370         STA $D201
403A- 8D 01 D3    1380         STA $D301
403D- C8          1390         INY              Y=1
403E- 8C 02 D2    1400         STY $D202    POINT TO 3RD CATALOG SECTOR
4041- C8          1410         INY              Y=2
4042- 8C 02 D3    1420         STY $D302    POINT TO 2ND CATALOG SECTOR
                  1430 *---FINISH THE VTOC--------------
4045- A0 00       1440         LDY #0       USE TABLES FOR VTOC
4047- BE 59 40    1450 .3      LDX TBLX,Y   INDEX INTO VTOC
404A- F0 09       1460         BEQ .4       ...FINISHED
404C- B9 68 40    1470         LDA TBLA,Y
404F- 9D 00 D0    1480         STA $D000,X
4052- C8          1490         INY
4053- D0 F2       1500         BNE .3       ...ALWAYS
                  1510 *-------------------------------
4055- AD 82 C0    1520 .4      LDA $C082    BACK TO MOTHERBOARD ROM
4058- 60          1530         RTS
                  1540 *-------------------------------
4059- 01 02 27
405C- 34 35 37
405F- 3C 3D 40
4062- 41 44 45
4065- 48 49 00    1550 TBLX    .HS 01.02.27.34.35.37.3C.3D.40.41.44.45.48.49.00
4068- 11 03 7A
406B- 23 10 01
406E- FF F0 FF
4071- FF FF FF
4074- FF FF       1560 TBLA    .HS 11.03.7A.23 10.01.FF.F0.FF.FF.FF.FF.FF.FF
                  1570 *-------------------------------
                  1580 RAMDISK.IMAG .PH $300
                  1590 RAMDISK.REAL
                  1600 *-------------------------------
0300-             1610 OLD.BD00    .BS 3
0303- 4C 06 03    1620 NEW.BD00    JMP LITTLE.RAM.DISK
                  1630 *-------------------------------
                  1640 LITTLE.RAM.DISK
0306- 84 48       1650         STY IOB
0308- 85 49       1660         STA IOB+1
030A- A0 01       1670         LDY #1       LOOK AT SLOT NUMBER
030C- B1 48       1680         LDA (IOB),Y
030E- C9 30       1690         CMP #$30     RAMDISK IN SLOT 3
0310- F0 1E       1700         BEQ RAM.DISK.SELECTED
0312- A0 02       1710         LDY #2
0314- B9 00 03    1720 .1      LDA OLD.BD00,Y
0317- 99 00 BD    1730         STA $BD00,Y
031A- 88          1740         DEY
031B- 10 F7       1750         BPL .1
031D- A4 48       1760         LDY IOB
031F- A5 49       1770         LDA IOB+1
0321- 20 00 BD    1780         JSR $BD00
0324- A0 02       1790         LDY #2
0326- B9 03 03    1800 .2      LDA NEW.BD00,Y
0329- 99 00 BD    1810         STA $BD00.Y
032C- 88          1820         DEY
032D- 10 F7       1830         BPL .2
032F- 60          1840         RTS
                  1850 *-------------------------------
                  1860 RAM.DISK.SELECTED
0330- C8          1870         INY              LOOK AT DRIVE
0331- B1 48       1880         LDA (IOB),Y
0333- 49 01       1890         EOR #1       MUST BE DRIVE 1
0335- D0 2F       1900         BNE .99      ...NOT DRIVE 1. ERROR
0337- 85 3C       1910         STA RAMP     LO-BYTE OF RAMPAGE
                  1920 *-------------------------------
```

```
0339- A0 05     1930          LDY #5        GET SECTOR #
033B- B1 48     1940          LDA (IOB),Y
033D- C9 10     1950          CMP #16
033F- B0 25     1960          BCS .99       BAD T/S
0341- 85 3D     1970          STA RAMP+1
0343- 88        1980          DEY           GET TRACK #
0344- B1 48     1990          LDA (IOB),Y
0346- F0 1E     2000          BEQ .99       INVALID TRACK #
0348- C9 11     2010          CMP #$11      IS IT VTOC TRACK?
034A- D0 0C     2020          BNE .2        NOT TRACK 17
                2030 *---TRACK 17----------------------
034C- A5 3D     2040          LDA RAMP+1          GET SECTOR #
034E- C9 04     2050          CMP #4              MUST BE 0-3
0350- B0 14     2060          BCS .99             NOT VALID T/S
0352- 09 D0     2070 .1       ORA #$D0      FORM HI-BYTE OF ADDRESS
0354- A2 00     2080          LDX #0        C083 BANK
0356- F0 20     2090          BEQ .4        ...ALWAYS
                2100 *---TRACK 1-4---------------------
0358- C9 05     2110 .2       CMP #5        OTHERWISE MUST BE TRACK 1-4
035A- B0 0A     2120          BCS .99       NOT VALID T/S
035C- C9 01     2130          CMP #1        TRACK 1?
035E- D0 0E     2140          BNE .3        ...NO
0360- A5 3D     2150          LDA RAMP+1          GET SECTOR #
0362- C9 04     2160          CMP #4              MUST BE 4-F
0364- B0 EC     2170          BCS .1              ...GOOD
                2180 *---ERROR-------------------------
0366- A0 0D     2190 .99      LDY #13
0368- A9 40     2200          LDA #$40
036A- 91 48     2210          STA (IOB),Y
036C- 38        2220          SEC
036D- 60        2230          RTS
                2240 *---------------------------------
036E- 0A        2250 .3       ASL           CHANGE 2.3,4 TO 20,30.40
036F- 0A        2260          ASL
0370- 0A        2270          ASL
0371- 0A        2280          ASL
0372- 69 B0     2290          ADC #$B0      ... TO D0-E0-F0
0374- 05 3D     2300          ORA RAMP+1          MERGE SECTOR
0376- A2 08     2310          LDX #8        C08B BANK
0378- 85 3D     2320 .4       STA RAMP+1
                2330 *---------------------------------
037A- A0 0C     2340          LDY #12       LOOK AT OPCODE
037C- B1 48     2350          LDA (IOB),Y
037E- F0 E6     2360          BEQ .99       ...NOT RD OR WRT
0380- C9 03     2370          CMP #3        IS IT RD OR WRT?
0382- B0 E2     2380          BCS .99       ...NO. IGNORE
0384- 4A        2390          LSR           SET CARRY IF READ, CLR IF WRT
                2400 *---SELECT RAMCARD BANK-----------
0385- BD 83 C0  2410          LDA $C083,X
0388- BD 83 C0  2420          LDA $C083,X
                2430 *---CLEAR ERROR CODE--------------
038B- A0 0D     2440          LDY #13
038D- A9 00     2450          LDA #0
038F- 91 48     2460          STA (IOB),Y
                2470 *---GET BUFFER ADDRESS------------
0391- A0 08     2480          LDY #8
0393- B1 48     2490          LDA (IOB),Y
0395- 85 3E     2500          STA BUFP
0397- C8        2510          INY
0398- B1 48     2520          LDA (IOB),Y
039A- 85 3F     2530          STA BUFP+1
039C- A0 00     2540          LDY #0
039E- B0 09     2550          BCS .6        ...READ
                2560 *---WRITE A SECTOR----------------
03A0- B1 3E     2570 .5       LDA (BUFP),Y
03A2- 91 3C     2580          STA (RAMP),Y
03A4- C8        2590          INY
03A5- D0 F9     2600          BNE .5
03A7- F0 07     2610          BEQ .7        ...ALWAYS
                2620 *---READ A SECTOR-----------------
03A9- B1 3C     2630 .6       LDA (RAMP),Y
03AB- 91 3E     2640          STA (BUFP),Y
03AD- C8        2650          INY
03AE- D0 F9     2660          BNE .6
03B0- AD 82 C0  2670 .7       LDA $C082     BACK TO MOTHERBOARD ROM
03B3- 18        2680          CLC
03B4- 60        2690          RTS
                2700 *---------------------------------
                2710          .EP
```

NEW PRODUCT



IIc SYSTEM CLOCK

- Fully ProDos compatible
- Automatic time and date stamping
- Easy to use from BASIC
- Battery operated, uses 3 "AA" batteries (will last 1-2 years before simple replacement)
- Date has year, month, date and day of week
- Time has hours, minutes and seconds
- Will time and date stamp AppleWorks files
- Will display time and date on the AppleWorks screen
- Auto access from AppleWorks data-base (just use a time and date field)
- Pass through serial port - The IIc system clock can plug into either the modem or printer serial port, then modem or printer plugs into the clock
- No hassle 5 year warranty
- Only $79.00

"We Set the Standard"

214-241-6060 APPLIED ENGINEERING 9 AM - 11 PM

Kablit Security System

After three burglaries or attempts here at the office, and four at Bill's house, we have been looking into ways to make our Apples a little more secure. There are a variety of products available these days, some involving special furniture that locks up around computer. and others consisting of brackets that lock the equipment to the desk top. These solutions seem too expensive and limiting for out purposes. We are always shifting the computers and monitors around to install or remove cards or to connect or disconnect some accessory. And four computers here in the office and two more at our houses mean that the system had better be inexpensive.

Well we have found what looks to be the answer: the Kablit Security System. from Secure-It, Inc. This is 10 feet of 3/16" steel cable with a high-quality padlock-type lock and an assortment of special hardware to attach the cable to your computer, monitor, disk drives, printer. or whatever. The connectors attach using the normal case screws of your equipment, so in most cases there is no need to drill holes or otherwise tear things up. There are specific kits for the Apple //c and the Macintosh.

The list price of the Kablit Security System is $49.95; we will be offering them for $45 + shipping.

An Easier QUIT from ProDOS.....Mark Jackson
Chicago, IL

When using a hard disk with ProDOS it is often useful to use the MLI QUIT call to go from one application to another. However, if you are deep within a subdirectory the QUIT code makes you retype the entire Prefix if you want to shorten it. To allow the use of the right arrow during the QUIT call do the following:

```
UNLOCK PRODOS
BLOAD PRODOS,A$2000,TSYS
CALL-151
5764:75       (for ProDOS 1.1.1 -- use 5964 for 1.0.1)
BSAVE PRODOS,A$2000,TSYS
LOCK PRODOS
```

This changes the input call to $FD75 which allows right arrow input. There is one drawback: now to restore the prompted prefix you must press ESCape when asked for the Pathname of the next application.

Solutions to Adam Levin's Painting Puzzle....Adam Levin, et al

The puzzle, published last month, was to write a program which would fill all RAM from $0000 through $BFFF with the same value. What value is your choice.

The listing of my solution follows. It executes at $9966, which is inside the middle DOS buffer. To get it there, you can BLOAD or BRUN it. A few seconds after the screen fill's up with "Y" characters, the program has completely filled RAM from $0000 through $BFFD with $99.

Lines 1080-1200 fill all the RAM not occupied by my program (addresses $0000-$98FF and $99C8-$BFFF) with $99. I first fill the RAM from $99C8 up, and then from $0000 up through $98FF. You have to forgive the self-modifying code in a puzzle solution like this.

Lines 1210-1280 store a NOP and a JMP $0000 at the end of RAM. Lines 1320-1350 store $99 into $9900-$9999. It's getting hot in here!

Lines 1390-1590 get executed more than once. The first time, they store $99 into $999B-$99A3, and $99A5. By this time every byte from $0000 through $99A5 is set $99. All those bytes can be executed as "STA $9999,Y" instructions, and the JMP $0000 we placed at the end of RAM will do just that. When we get back up to line 1430, at $99A9, we start moving Y again and store $99 into $99A6-99AC and $99AE. It progressively keeps covering itself up, and eventually it is all gone:

```
999B
999C
999D 99A6
999E 99A7 99AF
999F 99A8 99B0
99A0 99A9 99B1 99B7
99A1 99AA 99B2 99B8
99A2 99AB 99B3 99B9 99BD
99A3 99AC 99B4 99BA 99BE 99C1
...
99A5 99A3 99B6 99BC 99C0 99C3 99C5 99C6
```

```
                1000 *SAVE S.RAMFILL ADAM
                1010 *-------------------------------
                1020 *   ADAM LEVIN'S SOLUTION TO THE PUZZLE
                1030 *-------------------------------
                1040         .OR $9966     MUST START HERE
                1050         .TF B.PAINTER
                1060 *-------------------------------
                1070 PAINTER
9966- A0 C6     1080         LDY #END-1
9968- A9 99     1090         LDA #$99      STORE $99 FROM END OF PROGRAM
996A- 99 00 99  1100 COAT1   STA $9900,Y   THROUGH $BFFF
996D- C8        1110         INY
996E- D0 FA     1120         BNE COAT1
9970- EE 6C 99  1130         INC COAT1+2   NEXT PAGE
9973- AE 6C 99  1140         LDX COAT1+2
9976- E0 C0     1150         CPX #$C0      REACHED $BFFF YET?
9978- D0 05     1160         BNE .2        ...NOT YET
997A- A2 00     1170         LDX #0        WRAP AROUND AND STORE FROM
997C- 8E 6C 99  1180         STX COAT1+2   $0000 THRU $HERE
```

DISASM 2.2e - AN INTELLIGENT DISASSEMBLER : $30.00

Investigate the inner workings of machine language programs. DISASM converts machine code into meaningful, symbolic source. Creates a standard text file compatible with S-C, LISA, ToolKit and other assemblers. Handles data tables, displaced object code & even lets you substitute your own meaningful labels. (100 commonly used Monitor and Pg Zero names included.) An address-based triple cross reference table is provided to screen or printer. DISASM is an invaluable machine language learning aid to both novice & expert alike. Don Lancaster says DISASM is "absolutely essential" in his new ASSEMBLY COOKBOOK. For entire Apple II family including the new Apple //c (with all the new opcodes). SOURCE CODE available for an additional $30.00

LOW LOW PRICE ! ! ! C-PRINT For The APPLE //c : $69.00

Connect standard parallel printers to an Apple //c. C-PRINT is a hardware accessory that plugs into the standard Apple //c printer serial port. The other end plugs into any printer having a standard 36 pin centronics-type parallel connector. Just plug in and print! High speed data transfer at 9600 Baud. No need to reconfigure serial port or load software drivers for text printing.

FONT DOWNLOADER & EDITOR : $39.00

Turn your printer into a custom typesetter. Downloaded characters remain active while printer is powered. Use with any Word Processor program capable of sending ESC and control codes to printer. Switch back and forth easily between standard and custom fonts. All special printer functions (like expanded, compressed etc.) apply to custom fonts. Full HIRES screen editor lets you create your own characters and special graphics symbols. Compatible with many parallel printer I/F cards. User driver option provided. For Apple II, II+, //e. Specify printer: Apple Dot Matrix, C.Itoh 8510A (Prowriter), Epson FX 80/100, or OkiData 92/93.

The Font Downloader & Editor for the Apple Imagewriter Printer. For use with Apple II, II+, //e (with SuperSerial card) and the new Apple //c (with builtin serial interface).

FONT LIBRARY DISKETTE #1 : $19.00 Contains lots of user-contributed fonts for all printers supported by the Font Downloader & Editor. Specify printer with order.

The 'PERFORMER' CARD : $39.00

Plugs into any slot to convert a 'dumb' centronics-type printer I/F card into a 'smart' one. Command menu eliminates need to remember complicated ESC codes. Features include perforation skip, auto page numbering with date & title. Includes large HIRES graphics & text screen dumps. Specify printer: MX-80 with Graftrax-80, MX-100, MX-80/100 with Graftraxplus, NEC 8092A, C.Itoh 8510 (Prowriter), OkiData 82A/83A with Okigraph & OkiData 92/93. SOURCE CODE: $30.00

FIRMWARE FOR APPLE-CAT: The 'MIRROR' ROM : $25.00

Communications ROM plugs directly into Novation's Apple-Cat Modem card. Basic modes: Dumb Terminal, Remote Console & Programmable Modem. Features include: selectable pulse or tone dialing, true dialtone detection, audible ring detect, ring-back, printer buffer, 80 col card & shift key mod support. Uses superset of Apple's Comm card and Micromodem II commands. SOURCE CODE: $50.00

RAM/ROM DEVELOPMENT BOARD : $30.00

Plugs into any Apple slot. Holds one user-supplied 2Kx8 memory chip (6116 type RAM for program development or 2716 EPROM to keep your favorite routines on-line). Maps into $Cn00-CnFF and $C800-CFFF.

ALL NEW ! ! ! MIDI *MUSIC PRODUCTS*

MIDI means Musical Instrument Digital Interface. Use your computer with any MIDI-equipped music keyboard for entertainment and music education. Low cost MIDI player interface cable, complete with 6 song demo disk: $49.00. Thousands of popular songs available soon on diskette (also compatible with Passport MIDI interface). Products for both the Apple IIc and Commodore 64/128. Unique general purpose MIDI expander cable and gender changer also available. Send SASE for product descriptions and prices.

Avoid a $3.00 handling charge by enclosing full payment with order. VISA/MC and COD phone orders accepted.

RAK-WARE 41 Ralph Road W. Orange N J 07052 (201) 325-1885

```
997F- E0 99     1190 .2     CPX #$99      HAVE WE COME FULL CIRCLE?
9981- D0 E7     1200        BNE COAT1     ...NO, KEEP PAINTING
9983- A0 EA     1210        LDY #$EA      ...YES, NOW PATCH END OF RAM
9985- 8C FB BF  1220        STY $BFFB     FOR WRAPPING AROUND
9988- 8C FC BF  1230        STY $BFFC
998B- A0 4C     1240        LDY #$4C      NOP, JMP $0000
998D- 8C FD BF  1250        STY $BFFD
9990- A0 00     1260        LDY #0
9992- 8C FE BF  1270        STY $BFFE
9995- 8C FF BF  1280        STY $BFFF
                1290 *-------------------------------
                1300 *   PAINT $9900-HERE
                1310 *-------------------------------
9998- 99 00 99  1320 COAT2  STA $9900,Y
999B- C8        1330        INY
999C- C0 9A     1340        CPY #COAT2+2
999E- 90 F8     1350        BCC COAT2
                1360 *-------------------------------
                1370 *   TRY TO GET OUT WITHOUT LEAVING FOOTPRINTS!
                1380 *-------------------------------
99A0- A0 02     1390        LDY #2        SET INDEX TO POINT TO $999B
99A2- 99 99 99  1400        STA $9999,Y
99A5- C8        1410        INY           $999C
99A6- 99 99 99  1420        STA $9999,Y
99A9- C8        1430        INY           $999D
99AA- 99 99 99  1440        STA $9999,Y
99AD- C8        1450        INY           $999E
99AE- 99 99 99  1460        STA $9999,Y
99B1- C8        1470        INY           $999F
99B2- 99 99 99  1480        STA $9999,Y
99B5- C8        1490        INY           $99A0
99B6- 99 99 99  1500        STA $9999,Y
99B9- C8        1510        INY           $99A1
99BA- 99 99 99  1520        STA $9999,Y
99BD- C8        1530        INY           $99A2
99BE- 99 99 99  1540        STA $9999,Y
99C1- C8        1550        INY           $99A3
99C2- 99 99 99  1560        STA $9999,Y
99C5- C8        1570        INY           $99A4
99C6- C8        1580        INY           $99A5
99C7- 99 99 99  1590 END    STA $9999,Y
                1600 *-------------------------------
```

Bob S-C's solution

The program loads at $800, but actually executes at $100. Lines 1030-1080 move the filler program down to $100 and jump to it. This solution fills all of RAM from $0000-BFFF with $48, which is a "PHA" instruction.

To keep from running off the end of RAM into the I/O space, I took advantage of the fact that the keyboard register can be read at both $C000 and $C001. Lines 1140-1160 wait until you type a zero key ("0"). The ASCII code for "0" is $B0. Two $B0 values in a row at $C000 and $C001 will dis-assemble as a BCS to $BFB2. Hence my solution finishes with an infinite loop running from $BFB2 to $C001.

Lines 1170-1290 fill RAM from $200-$BFFF with $48's, which are "PHA" opcodes. Lines 1300-1330 do the same with page zero.

Line 1350 jumps to $200, which means that the PHA opcodes start being executed. Since the stack is only 256 bytes long, and since the stack pointer wraps around, by the time the PHA at $2FF has executed all of page 1 will have been filled with $48. Since carry is set, when execution reaches $C000 the processor will go into that infinite loop I mentioned above.

```
                    1000 *SAVE S.RAMFILL RBSC
                    1010 *-------------------------------
                    1020         .OR $800
0800- A0 27         1030 MOVER   LDY #LENGTH        MOVE "FILLER" PROGRAM
0802- B9 0E 08      1040 .1      LDA MY.FILLER,Y    TO EXECUTION AREA
0805- 99 00 01      1050         STA FILLER,Y       AT $100...
0808- 88            1060         DEY
0809- 10 F7         1070         BPL .1
080B- 4C 00 01      1080         JMP FILLER         NOW START FILLING!
                    1090 *-------------------------------
                    1100 *    FOLLOWING CODE EXECUTES AT $100...
                    1110 *-------------------------------
                    1120 MY.FILLER .PH $100
                    1130 FILLER
0100- AD 00 C0      1140 .1      LDA $C000     WAIT UNTIL "0" TYPED
0103- C9 B0         1150         CMP #$B0      ($B0 IS ALSO BCS OPCODE)
0105- D0 F9         1160         BNE .1
                    1170 *---FILL $200-$BFFF--------------
0107- A0 00         1180         LDY #0
0109- 84 00         1190         STY 0
010B- A9 02         1200         LDA #2
010D- 85 01         1210         STA 1
010F- A9 48         1220         LDA #$48      PHA OPCODE
0111- 91 00         1230 .2      STA (0),Y
0113- C8            1240         INY
0114- D0 FB         1250         BNE .2
0116- E6 01         1260         INC 1
0118- A6 01         1270         LDX 1
011A- E0 C0         1280         CPX #$C0      UNTIL $BFFF
011C- 90 F3         1290         BCC .2
                    1300 *---FILL PAGE ZERO---------------
011E- 99 00 00      1310 .3      STA 0,Y
0121- C8            1320         INY
0122- D0 FA         1330         BNE .3
                    1340 *---FILL PAGE ONE----------------
0124- 4C 00 02      1350         JMP $200
27-                 1360 LENGTH  .EQ *-FILLER
                    1370         .EP
                    1380 *-------------------------------
```

David Johnson's solution

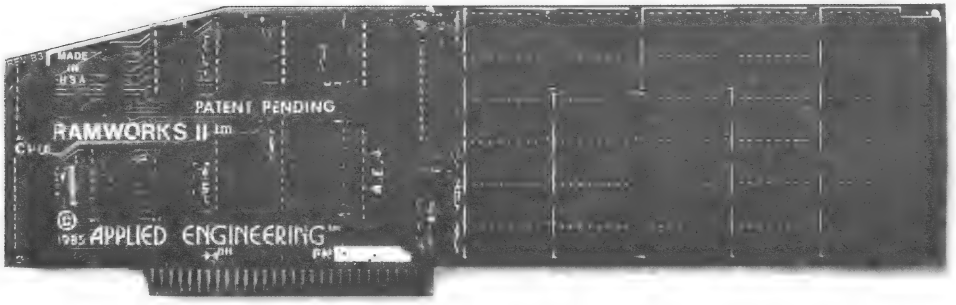
My solution uses the power of the 65802. There was no restriction to the 6502 mentioned in the puzzle last month. All 49152 locations of motherboard RAM are filled with $DB, which happens to be the opcode value for the "STP" opcode. STP means "stop the processor", so once all RAM is filled it quits!

I use the MVP instruction to do the actual filling. The MVP instruction is located at $0000. I first put $DB into $BFFF. Then I set up the registers so that MVP will copy $BFFF into $BFFE, then $BFFE into $BFFD, and so on down to copying $0001 into $0000. By this time the MVP runs out, and the processor executes the STP opcode at $0003.

The 2nd and 3rd bytes of the MVP opcode specify which 64K memory banks to use; on a 65802 these don't do anything, because the bank addresses don't get out of the chip. On a 65816 my program won't work correctly, because the bank bytes will be changed at the end. First the Source bank address will be changed, so that a byte will be copied from $DB.0002 into $00.0001. Now the Destination Bank Address is changed, to we don't know what: we will finally copy $DB.0001 into $xx.0000. That last byte-move could be catastrophic (who knows, since we don't have any 65816-based systems yet?). Anyway, my program works fine in an Apple equipped with a 65802.

Meet RamWorks II®

The Recognized Industry Standard For Memory Expansion of the Apple IIe.



RamWorks II. A Generation Ahead. Again.

The best selling expansion card for the Apple IIe just got even better. With RamWorks II, expand your IIe to an incredible 3 megabytes of usable RAM.

Turbo Charged AppleWorks.

RamWorks II plugs into the IIe auxiliary slot and acts just like Apple's extended 80 column card, only better—because if you buy a 256K or larger card, AppleWorks will automatically load itself into RamWorks II. This dramatically increases AppleWorks' speed and power because it effectively eliminates the time required to access disk drive 1. Now, switch from word processing to spreadsheet to database management at the speed of light. AppleWorks responds the moment your fingers touch the keyboard.

But AppleWorks has certain internal limits, independent of available memory. Fear not. Only RamWorks II (and the original RamWorks of course) removes those limits. Only RamWorks II increases the maximum number of records available from 1,350 to over 16 000. Only RamWorks II actually increases the number of lines permitted in the word processing mode. And only RamWorks II features a built-in printer buffer, so you no longer have to wait for your printer to stop before going back to AppleWorks (256K or larger RamWorks II required).

With RamWorks II, you won't have to split your data into 2 or more separate files because you'll have the necessary memory to access ALL your data ALL the time, quickly and conveniently.

RamWorks II	AppleWorks Desktop
128K	101K
256K	188K
512K	378K
1 MEG	758K
1.5 MEG	1136K
3 MEG	2277K

The Most Friendly, Most Expandable Card Available.

RamWorks II is compatible with more off-the-shelf software than any other RAM card. Popular programs like Advanced VisiCalc, Magic Office System, Flashcalc, The Spread Sheet, Diversi-DOS, Supercalc 3A, Magicalc, etc. (and hardware add-ons like Profile and Sider hard disks). Fact is, only RamWorks is 100% compatible with all software written for the Apple 80 column and extended 80 column cards. In addition, RamWorks II can emulate most other RAM cards, so you can use programs written for them without modification. And any size RamWorks II can be user upgraded later to any larger size.

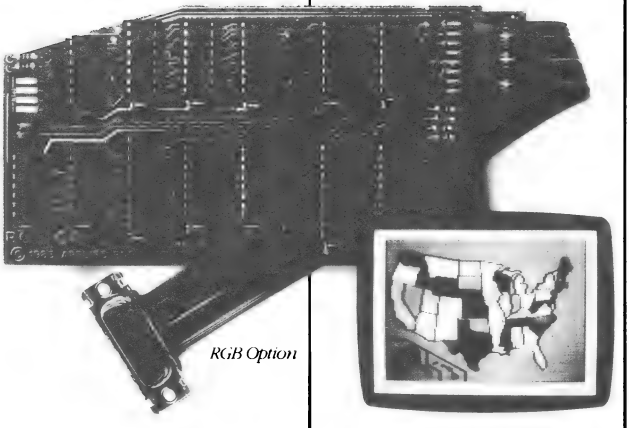
RamWorks II was designed so you could take full advantage of future developments in 16 and 32 bit microprocessors. As your needs grow, so can RamWorks II. A handy coprocessor connector allows the latest and greatest coprocessor cards to access all 3 MEG

of RamWorks II memory. And speaking of more memory, RamWorks II has a memory expansion connector on board so a low profile (no slot 1 interference) memory expansion card can add another 512K or 2 MEG of memory.

Unlike Apple's smaller, more expensive RAM card, RamWorks II plugs into the IIe auxiliary slot and therefore leaves slots 4 and 5 available for other peripheral cards.

It's In Color

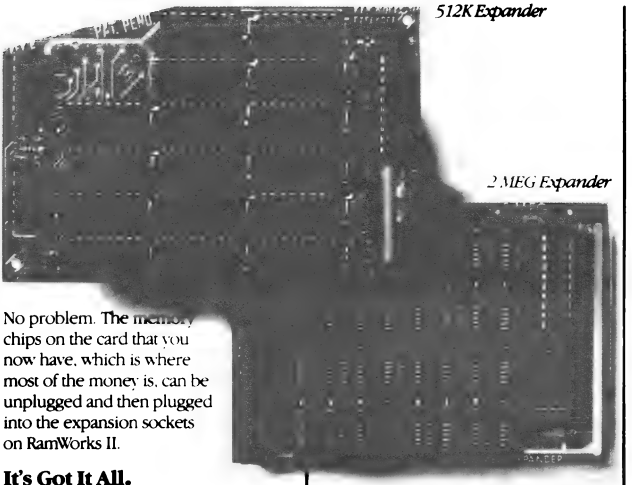
RamWorks II by itself is *fully* compatible with both the Apple monochrome and color monitors. But if you want better color graphics *plus* a more readable 80 column text (that blows away any composite color monitor) you'll appreciate our RGB color option. For only $129, it can be added to RamWorks II, giving you a razor sharp, vivid brilliance that's unsurpassed in the industry. The RGB option does not waste another valuable slot, but rather plugs into the back of RamWorks II with no slot 1 interference (works on the original RamWorks, too) and attaches to virtually any RGB monitor. And remember. You can order the RGB option with your RamWorks II. Or add it on at a later date.



RGB Option

It Corrects Mistakes.

Let's say you bought some other RAM card (and that's a mistake) and your RAM card is not being recognized by AppleWorks, Advanced Visicalc, Flashcalc, Supercalc 3A, or other programs, and you want RamWorks II.



512K Expander

2 MEG Expander

No problem. The memory chips on the card that you now have, which is where most of the money is, can be unplugged and then plugged into the expansion sockets on RamWorks II.

It's Got It All.

- 15 Day Money Back Guarantee
- Super sharp 80 column text (patent pending) with or without RGB option
- Double high resolution graphics (with or without RGB option)
- Expandable up to 1 Meg (1024K) on main board
- Expandable to 3 Meg (3072K) with expander (piggyback) card
- Can use 64K or 256K RAMS in any combination
- Linear addressing coprocessor port
- Automatic AppleWorks expansion up to 227?K desktop
- Accelerates AppleWorks
- Built-in AppleWorks printer buffer
- The only large RAM card that's 100% compatible with all IIe software
- RamDrive™ the ultimate disk emulation software included free
- 16 Bit option
- Compatible RGB option
- Built-in self diagnostics software
- No slot 1 interference
- Lowest power consumption (patent pending)
- Takes only one slot (auxiliary)
- Software industry standard
- Advanced Computer Aided Design
- Used by Apple Computer, Steve Wozniak and virtually all software companies
- Displays date and time on the AppleWorks screen with any PRO-DOS compatible clock
- 5 Year no hassle warranty

RamWorks II with 64K	**$ 179**
RamWorks II with 256K	**$ 219**
RamWorks II with 512K	**$ 269**
RamWorks II with 1 MEG	**$ 389**
RamWorks II with 1.5 MEG	**$ 549**
RamWorks II with 3 MEG	**$1699**
RGB Option (may add later)	**$ 129**
16 Bit Option (may add later)	**$ 89**

RamWorks II. The industry standard for memory expansion of the Apple IIe.

ORDER YOUR RamWorks II TODAY. 9 a.m. to 11 p.m. 7 Days, or send check or money order to Applied Engineering.

MasterCard, Visa and C.O.D. welcome. Texas Residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

```
                    1000 *SAVE DAVID JOHNSON'S FILLER
                    1010 *-------------------------------
                    1020 *      SOLUTION TO PUZZLE BY DAVID C. JOHNSON
                    1030 *-------------------------------
                         1040         .OP 65802     I got mine!
                         1050 *-------------------------------
                         1060         .OR $00
                         1070 *-------------------------------
000000- 44 00 00         1080 paint    mvp 0.0       fill $BFFE-$0000 from $BFFF
                         1090 *-------------------------------
000003- A9 DB            1100 START    LDA #$DB      "STP" OPCODE
000005- 8D FF BF         1110          STA $BFFF     SEED FOR THE "MVP" INSTRUCTION
000008- 18               1120          CLC           GET INTO NATIVE MODE
000009- FB               1130          XCE
00000A- C2 30            1140          REP #$30      16-BIT REGISTERS
00000C- A2 FF BF         1150          LDX ##$BFFF   Source Address = $BFFF
00000F- 9B               1160          TXY
000010- 88               1170          DEY           Destination Address = $BFFE
000011- 98               1180          TYA           # Bytes -1 to be "moved"
000012- 80 EC            1190          BRA paint     MVP must be at $0000
                         1200 *-------------------------------
```

Using the Object Vector in S-C Macro Assembler.....Bill Morgan

Sometimes we want to do something special with the object code generated by the S-C Macro Assembler. Maybe write it directly into an EPROM programmer, send it out through a serial port, or store it into some special device. One such device is the Douglas Electronics Writable ROM Board, which appears to the Apple as 2K of RAM at $C800 but brings out a cable that plugs right into a 2716 EPROM socket. With this card we can test the assembled code instantly in the target machine, without the delay and hassle of programming and transferring an EPROM.

There are a couple of hitches along the way. The assembler normally protects everything above $BFFF from code storage, and we need some special code because we have to temporarily switch off any other card using $C800, switch on the WROM Board, write a byte, and switch the WROM Board off again.

Fortunately, Version 2.0 of the S-C Macro Assembler has some special features for cases just like this. There are parameters at the beginning of the assembler to unprotect a specified area of memory, and each byte generated is passed through an Object Vector on its way to storage, so we can intercept the byte and do our memory switching before passing it back to the assembler.

Since the object code is going to be stored in successive memory locations pointed to by the Target Address, we can just use the Macro Assembler's normal STORE.OBJECT.BYTE routine. The address of STORE.OBJECT.BYTE is in the JMP instruction at OBJECT.VECTOR, so it's easy to get that address, plug it into our code, and then install our address in OBJECT.VECTOR. If we needed to do something different with the object code, like storing each byte into the same hardware register, we would do that instead at the line labelled CALL.

Writable ROM Board, by Douglas Electronics, 718 Marina Blvd., San Leandro, CA 94577. (415) 483-8770. $95.

```
                        1000 *SAVE S.WROMWRITE
                        1010 *-------------------------------
01-                     1020 LC         .EQ 1       1 if $D000 assembler
07-                     1030 WROMSLOT   .EQ 7
                        1040 *-------------------------------
                        1050         .DO LC
D012-                   1060 OBJECT.VECTOR  .EQ $D012
D024-                   1070 UNPROTECT.LOW  .EQ $D024
D026-                   1080 UNPROTECT.HIGH .EQ $D026
                        1090         .ELSE
                        1130         .FIN
                        1140
C0F0-                   1150 WRITECARD    .EQ WROMSLOT*$10+$C080
C0F1-                   1160 CARDOFF      .EQ WROMSLOT*$10+$C081
CFFF-                   1170 C800.OFF     .EQ $CFFF
                        1180
C800-                   1190 TARGET.LOW   .EQ $C800
CFFF-                   1200 TARGET.HIGH  .EQ $CFFF
                        1210 *-------------------------------
                        1220          .OR $300
                        1230          .TF WROMWRITE
                        1240 INSTALL
                        1250         .DO LC
0300- 2C 83 C0          1260           BIT $C083
0303- 2C 83 C0          1270           BIT $C083
                        1280         .FIN
0306- A9 C8             1290           LDA /TARGET.LOW
0308- 8D 25 D0          1300           STA UNPROTECT.LOW+1
030B- A9 00             1310           LDA #TARGET.LOW
030D- 8D 24 D0          1320           STA UNPROTECT.LOW
0310- A9 CF             1330           LDA /TARGET.HIGH
0312- 8D 27 D0          1340           STA UNPROTECT.HIGH+1
0315- A9 FF             1350           LDA #TARGET.HIGH
0317- 8D 26 D0          1360           STA UNPROTECT.HIGH
031A- AD 14 D0          1370           LDA OBJECT.VECTOR+2
031D- 8D 3C 03          1380           STA CALL+2
0320- AD 13 D0          1390           LDA OBJECT.VECTOR+1
0323- 8D 3B 03          1400           STA CALL+1
0326- A9 03             1410           LDA /CARDON
0328- 8D 14 D0          1420           STA OBJECT.VECTOR+2
032B- A9 34             1430           LDA #CARDON
032D- 8D 13 D0          1440           STA OBJECT.VECTOR+1
                        1450         .DO LC
0330- 2C 80 C0          1460           BIT $C080
                        1470         .FIN
0333- 60                1480           RTS
                        1490 *-------------------------------
0334- 2C FF CF          1500 CARDON BIT C800.OFF
0337- 2C F0 C0          1510        BIT WRITECARD
033A- 20 FF FF          1520 CALL   JSR $FFFF
033D- 2C F1 C0          1530        BIT CARDOFF
0340- 60                1540        RTS
```

Note on Mainstay MACASM for the Macintosh

We still have a small supply of the original release of this highly-praised development tool for the Macintosh. (Even Jerry Pournelle had good words for it.) I say original edition, because they are now at version 1.2, with 1.3 scheduled in January.

Mainstay has told us that there is little real difference in the various versions, not enough to influence your decision as to where to buy. And they also have a policy that they will provide your first upgrade absolutely free. All you need to do is fill in your registration card, make a backup copy of MacASM to use in the interim, and send them your original MacASM disk.

Their current end-user price is $125. Note that ours are still being sold at the introductory price of $100. Wow! It's a steal!

Commented Listing of ProDOS QUIT Code.....Bob Sander-Cederlof

After reading Mark Jackson's article on improving the ProDOS QUIT code, I though it would be nice to have a commented listing of that program. The listing which follows is just that.

The ProDOS QUIT code is booted into $D100-D3FF in the alternate $D000 bank (the one you get by diddling $C083). Normally ProDOS MLI stays in the $C08B side. When a program issues the QUIT call (MLI code $65), the contents of $D100-D3FF are copied to $1000-12FF; then ProDOS jumps to $1000.

If you BLOAD the SYS file named PRODOS from a bootable ProDOS 1.1.1 disk, and examine it, you will find that it is laid out in eight parts. The first part is a relocator, which copies the other seven parts into their normal homes. Like this:

Position as loaded	Position copied to	
2000-29FF	---	Relocator
2A00-2BFF	Aux 200-3FF	/RAM/ driver
2C00-2C7F	FF00-FF7F	/RAM/ driver
2C80-2CFF	nowhere	All zeroes
2D00-4DFF	D000-F0FF	MLI Kernel
4E00-4EFF	BF00-BFFF	System Global Page
4F00-4F7F	D742-D7BD	Thunderclock driver
4F80-4FFF	FF80-FFFF	Interrupt Code
5000-56FF	F800-FEFF	Device Drivers
5700-59FF	D100-D3FF(alt)	QUIT Code
zeroes	F100-F7FF	

The part I am interested in right now is the QUIT code, which is at $5700-$59FF in the PRODOS file.

The QUIT code is not written very efficiently. For some reason, there are two completely separate editing programs: one for the prefix, and another for the pathname. (And as Mark points out, neither one is very handy.) Even the code that initializes the BITMAP is inefficient.

```
                1000 *SAVE S.PRODOS.QUIT
                1010 *--------------------------------
24-             1020 CH      .EQ $24
25-             1030 CV      .EQ $25
DE-             1040 ERRCOD  .EQ $DE
                1050 *--------------------------------
0280-           1060 BUF     .EQ $0280
                1070 *--------------------------------
2000-           1080 SYSTEM  .EQ $2000
                1090 *--------------------------------
BF00-           1100 MLI     .EQ $BF00
BF58-           1110 BITMAP  .EQ $BF58
                1120 *--------------------------------
C000-           1130 KEY     .EQ $C000
C000-           1140 S80STOREOFF .EQ $C000
C00C-           1150 S80OFF  .EQ $C00C
C00F-           1160 SALTON  .EQ $C00F
C010-           1170 STROBE  .EQ $C010
C082-           1180 ROM     .EQ $C082
                1190 *--------------------------------
```

1600k IIe!

Why settle for less when you can buy Checkmate Technology's **State-Of-The-Art MULTIRAM IIe™** from Coit Valley Computers with the following features:

- **DIRECT SUBSTITUTE FOR RAMWORKS™** or Apple Extended 80 column cards. Because **MULTIRAM** follows Apples' strict guidelines, it **RUNS ALL 3rd PARTY SOFTWARE** written for either card.
- ***UP TO 768k MAIN BOARD MEMORY - 50% MORE THAN RAMWORKS!***
- **UP TO 768k MORE OPTIONAL PIGGYBACK BOARD MEMORY + FREE RGB ON THE SAME BOARD** - and it won't interfere with cards in slot-1! **A TOTAL OF 1600K IIe MEMORY AT A LOWER PRICE!**
- **FREE RAM DISK, RAM TEST and APPLEWORKS™ EXPANDER SOFTWARE** that can automatically load AppleWorks entirely into memory, run 20 x faster, increase the Desktop to 1100k, auto-segment large files onto multiple disks, and store up to 5100 records!
- **FREE SOFTWARE UPDATES** exclusively at Coit Valley Computers.
- **TRUE 16 BIT CO-PROCESSOR PORT** with linear connected data banks needed for easy Co-Processor access.
- **EXCLUSIVE 5 YEAR WARRANTY THAT, UNLIKE RAMWORKS, INSURES COVERAGE NO MATTER WHERE YOU BOUGHT IT!**

AVAILABLE NOW AT LOWER COST AND A MONEY BACK SATISFACTION GUARANTEE from **Coit Valley Computers*.** For a limited time, we'll even give an **EXTRA 64k OF MEMORY FREE** with each 256k or 512k MULTIRAM IIe card! **CALL FOR CURRENT PRICING!**

	OUR LOW PRICE		OUR LOW PRICE
64k **MultiRam** IIe	**155.**	1536k **MultiRam** IIe	**699. /FREE RGB**
128k **MultiRam** IIe	**173.**		
320k **MultiRam** IIe	**225.**	64k Memory Expander Chips (8)	**25.**
576k **MultiRam** IIe	**285.**	256k Memory Expander Chips (8)	**75.**
768k **MultiRam** IIe	**345.**	Pico™ Slimline Drive IIc, IIe, II+	**178.**
1024k **MultiRam** IIe	**599./FREE RGB**	**Apple IIe Enhancement Kit**	**62.**
1280k **MultiRam** IIe	**654./FREE RGB**	Promodem 1200A Modem	**369.**

640k 16-BIT IIc!

Checkmate Technology's **State-Of-The-Art MULTIRAM IIc** can expand your Apple IIc up to 640k and has a **16-bit PROCESSOR** option. It comes with the same **FREE APPLEWORKS EXPANDER, Ram Disk, and Ram Test** software as **MULTIRAM IIe** and is available now from Coit Valley Computers. **CALL FOR CURRENT PRICING!**

- **LOW COST EASILY INSTALLED 16-BIT CO-PROCESSOR OPTION ($145)** - unavailable from the competition. 65816
- **NO JUMPER WIRES OR CLIPS REQUIRED FOR INSTALLATION** - unlike the competition.
- **ALL CHIPS ARE SOCKETED AND REMOVABLE** - unlike the competition.
- **USES ABOUT 50% LESS POWER** than the competition.
- **IMPROVED KEYBOARD SUPPORT** to avoid spongy feeling keys.
- **MONEY BACK SATISFACTION GUARANTEE** from Coit Valley Computers*.
- **5 YEAR WARRANTY THAT, UNLIKE THE COMPETITION, INSURES COVERAGE NO MATTER WHERE YOU BOUGHT IT!**

	OUR LOW PRICE		OUR LOW PRICE
256k **MULTIRAM** IIc	**299.**	**16-BIT CO-PROCESSOR KIT**	**145.**
512k **MULTIRAM** IIc	**369.**	(optional for either size) 65816	

Terms: For fastest delivery send Cashier's/Certified check, Money Order. C.O.D. (add $5) & personal checks accepted (allow 14 days). Add $4 shipping & phone # to all orders. Add 3% for P.O.'s & MasterCard/Visa (include #/expir). Tex res add 6⅛% tax. **CALL FOR LATEST PRICES!**

MultiRam, Ramworks, Appleworks, Pico, respective trademarks of Checkmate Technology, Applied Engineering, Apple Comp, WGE Int. *Return your MultiRam within 10 days of receipt in original condition and receive complete refund less $5.

COIT VALLEY COMPUTERS **(214) 234-5047**
14055 Waterfall Way Dallas, Texas 75240

```
FC58-                 1200 HOME   .EQ $FC58
FC9C-                 1210 CLREOL .EQ $FC9C
FD0C-                 1220 RDKEY  .EQ $FD0C
FD8E-                 1230 CROUT  .EQ $FD8E
FDED-                 1240 COUT   .EQ $FDED
FE89-                 1250 SETKBD .EQ $FE89
FE93-                 1260 SETVID .EQ $FE93
FF3A-                 1270 BELL   .EQ $FF3A
                      1280 *--------------------------------
                      1290         .MA MLI
                      1300         JSR MLI
                      1310         .DA #$]1,]2
                      1320         .EM
                      1330 *--------------------------------
                      1340         .OR $1000
                      1350         .TA $5700
                      1360 *--------------------------------
                      1370 PRODOS.QUIT
1000- AD 82 C0        1380         LDA ROM       TURN ON THE MONITOR ROM
1003- 20 93 FE        1390         JSR SETVID    GET BACK TO GOOD OLD-FASHIONED
1006- 20 89 FE        1400         JSR SETKBD         DOWN-HOME 40 COLUMN DISPLAY
1009- 8D 0C C0        1410         STA S80OFF
100C- 8D 0F C0        1420         STA SALTON    Know what I mean, Vern?
100F- 8D 00 C0        1430         STA S80STOREOFF
                      1440 *---PREPARE BITMAP----------------
1012- A2 17           1450         LDX #$17
1014- A9 01           1460         LDA #1        Mark $BFxx in use
1016- 9D 58 BF        1470         STA BITMAP,X
1019- CA              1480         DEX
101A- A9 00           1490         LDA #0        Most pages are free
101C- 9D 58 BF        1500 .1      STA BITMAP,X
101F- CA              1510         DEX
1020- 10 FA           1520         BPL .1
1022- A9 CF           1530         LDA #$CF      $0000-01FF, $0400-07FF in use
1024- 8D 58 BF        1540         STA BITMAP
                      1550 *---DISPLAY PREFIX----------------
                      1560 GET.PREFIX
1027- 20 58 FC        1570         JSR HOME
102A- 20 8E FD        1580         JSR CROUT
102D- A9 41           1590         LDA #Q.PRFX
102F- 8D E9 11        1600         STA MSG.ADDR
1032- A9 12           1610         LDA /Q.PRFX
1034- 8D EA 11        1620         STA MSG.ADDR+1
1037- 20 E6 11        1630         JSR PRINT.MESSAGE
103A- A9 03           1640         LDA #3        VTAB 4
103C- 85 25           1650         STA CV
103E- 20 8E FD        1660         JSR CROUT     MAKE IT 5
1041-                 1670         >MLI C7,PREFIX.PARM
1041- 20 00 BF  0000>                JSR MLI
1044- C7 F8 12  0000>                .DA #$C7,PREFIX.PARM
1047- AE 80 02        1680         LDX BUF       # CHARS IN PREFIX
104A- A9 00           1690         LDA #0        MARK END OF PREFIX WITH 00
104C- 9D 81 02        1700         STA BUF+1,X   SO OUR MESSAGE PRINTER WILL
104F- A9 81           1710         LDA #BUF+1         PRINT IT.
1051- 8D E9 11        1720         STA MSG.ADDR
1054- A9 02           1730         LDA /BUF+1
1056- 8D EA 11        1740         STA MSG.ADDR+1
1059- 20 E6 11        1750         JSR PRINT.MESSAGE
                      1760 *---GET NEW PREFIX----------------
105C- A2 00           1770         LDX #0
105E- C6 25           1780         DEC CV        MOVE CURSOR TO BEGINNING OF LINE
1060- 20 8E FD        1790         JSR CROUT
                      1800 NEXT.PREFIX.CHAR
1063- 20 0C FD        1810         JSR RDKEY
1066- C9 8D           1820         CMP #$8D
1068- F0 4E           1830         BEQ SET.NEW.PREFIX    ...ACCEPT WHAT IS ON SCREEN
106A- 48              1840         PHA           ERASE PREFIX FROM SCREEN
106B- 20 9C FC        1850         JSR CLREOL
106E- 68              1860         PLA
106F- C9 9B           1870         CMP #$9B      IS CHAR <ESCAPE>?
1071- F0 B4           1880         BEQ GET.PREFIX    ...YES, START ALL OVER
1073- C9 98           1890         CMP #$98      IS CHAR CTRL-X?
                      1900 START.PREFIX.OVER
1075- F0 B0           1910         BEQ GET.PREFIX     ...START ALL OVER
1077- C9 89           1920         CMP #$89      IS CHAR <TAB>?
1079- F0 13           1930         BEQ .3        ...YES, RING BELL
107B- C9 88           1940         CMP #$88      IS CHAR BACKSPACE?
107D- D0 0D           1950         BNE .2        ...NO, APPEND TO LINE
```

```
107F- E0 00     1960          CPX #0        ...BACKSPACE, UNLESS AT BEGINNING
1081- F0 03     1970          BEQ .1        AT BEGINNING ALREADY
1083- C6 24     1980          DEC CH        BACK UP
1085- CA        1990          DEX
1086- 20 9C FC  2000 .1       JSR CLREOL    CHOP OFF AFTER CURSOR
1089- 4C 63 10  2010          JMP NEXT.PREFIX.CHAR
108C- B0 06     2020 .2       BCS .4        OTHER CONTROL CHAR < $88
108E- 20 3A FF  2030 .3       JSR BELL
1091- 4C 63 10  2040          JMP NEXT.PREFIX.CHAR
1094- C9 DB     2050 .4       CMP #"Z"+1
1096- 90 02     2060          BCC .5        ...NOT LOWER CASE
1098- 29 DF     2070          AND #$DF      CONVERT LOWER CASE TO UPPER
109A- C9 AE     2080 .5       CMP #"."      ALLOW PERIOD, SLASH, DIGITS
109C- 90 F0     2090          BCC .3        ...TOO SMALL
109E- C9 DB     2100          CMP #"Z"+1    ALLOW LETTERS
10A0- B0 EC     2110          BCS .3        ...TOO LARGE
10A2- C9 BA     2120          CMP #"9"+1
10A4- 90 04     2130          BCC .6        ...PERIOD, SLASH, OR DIGIT
10A6- C9 C1     2140          CMP #"A"
10A8- 90 E4     2150          BCC .3        ...NOT A LEGAL CHARACTER
10AA- E8        2160 .6       INX
10AB- E0 27     2170          CPX #$27
10AD- B0 C6     2180          BCS START.PREFIX.OVER    ...TOO LONG
10AF- 9D 80 02  2190          STA BUF,X
10B2- 20 ED FD  2200          JSR COUT      ECHO THE CHARACTER
10B5- 4C 63 10  2210          JMP NEXT.PREFIX.CHAR
                2220 *--------------------------------
                2230 SET.NEW.PREFIX
10B8- E0 00     2240          CPX #0        DID WE CHANGE IT?
10BA- F0 12     2250          BEQ GET.PATHNAME    ...NO
10BC- 8E 80 02  2260          STX BUF             ...YES, SO TELL SYSTEM
10BF-           2270          >MLI C6,PREFIX.PARM
10BF- 20 00 BF  0000>          JSR MLI
10C2- C6 F8 12  0000>          .DA #$C6,PREFIX.PARM
10C5- 90 07     2280          BCC GET.PATHNAME    ...NO ERRORS
10C7- 20 3A FF  2290          JSR BELL      DING, DONG!
10CA- A9 00     2300          LDA #0        SET .EQ. STATUS
10CC- F0 A7     2310 PFXOVR   BEQ START.PREFIX.OVER    ...ALWAYS
                2320 *--------------------------------
                2330 GET.PATHNAME
10CE- 20 58 FC  2340          JSR HOME
                2350 START.PATHNAME.OVER
10D1- 20 8E FD  2360          JSR CROUT
10D4- A9 69     2370          LDA #Q.PATH
10D6- 8D E9 11  2380          STA MSG.ADDR
10D9- A9 12     2390          LDA /Q.PATH
10DB- 8D EA 11  2400          STA MSG.ADDR+1
10DE- 20 E6 11  2410          JSR PRINT.MESSAGE
10E1- A9 03     2420          LDA #3        VTAB 4
10E3- 85 25     2430          STA CV
10E5- 20 8E FD  2440          JSR CROUT     MAKE IT 5
10E8- A2 00     2450          LDX #0
                2460 NEXT.PATHNAME.CHAR
10EA- A9 FF     2470          LDA #$FF      CURSOR CHARACTER
10EC- 20 ED FD  2480          JSR COUT
10EF- C6 24     2490          DEC CH        BACK UP OVER CURSOR
10F1- AD 00 C0  2500 .1       LDA KEY
10F4- 10 FB     2510          BPL .1        ...WAIT TILL KEY PRESSED
10F6- 8D 10 C0  2520          STA STROBE
10F9- C9 9B     2530          CMP #$9B      <ESCAPE>?
10FB- D0 06     2540          BNE .2        ...NO
10FD- A5 24     2550          LDA CH        IF AT BEGINNING, GET PREFIX OVER
10FF- D0 CD     2560          BNE GET.PATHNAME       ...ELSE GET PATHNAME OVER
1101- F0 C9     2570          BEQ PFXOVR
1103- C9 98     2580 .2       CMP #$98      CONTROL-X?
1105- F0 C7     2590 .3       BEQ GET.PATHNAME
1107- C9 89     2600          CMP #$89      TAB KEY?
1109- F0 09     2610          BEQ .5        ...YES
110B- C9 88     2620          CMP #$88      BACKSPACE?
110D- D0 03     2630          BNE .4        ...NO
110F- 4C D0 11  2640          JMP BACKSPACE.IN.PATHNAME
                2650 *--------------------------------
1112- B0 06     2660 .4       BCS .6
1114- 20 3A FF  2670 .5       JSR BELL      ...INVALID CHAR, RING BELL
1117- 4C EA 10  2680          JMP NEXT.PATHNAME.CHAR
                2690 *--------------------------------
111A- C9 8D     2700 .6       CMP #$8D
111C- F0 29     2710          BEQ SET.NEW.PATHNAME
```

```
111E- C9 DB     2720         CMP #"Z"+1
1120- 90 02     2730         BCC .7
1122- 29 DF     2740         AND #$DF       CHANGE LOWER CASE TO UPPER
1124- C9 AE     2750 .7      CMP #"."       ACCEPT DOT, SLASH, OR DIGIT
1126- 90 EC     2760         BCC .5         ...TOO SMALL
1128- C9 DB     2770         CMP #"Z"+1     ACCEPT LETTERS
112A- B0 E8     2780         BCS .5         ...TOO LARGE
112C- C9 BA     2790         CMP #"9"+1
112E- 90 04     2800         BCC .8         ...DOT, SLASH, OR DIGIT
1130- C9 C1     2810         CMP #"A"
1132- 90 E0     2820         BCC .5         ...NOT A VALID CHARACTER
1134- 48        2830 .8      PHA            CLEAR BEYOND THIS POINT
1135- 20 9C FC  2840         JSR CLREOL
1138- 68        2850         PLA
1139- 20 ED FD  2860         JSR COUT       ECHO THE NEW CHARACTER
113C- E8        2870         INX
113D- E0 27     2880         CPX #$27
113F- B0 C4     2890         BCS .3         ...NAME TOO LONG
1141- 9D 80 02  2900         STA BUF,X      APPEND CHAR TO NAME
1144- 4C EA 10  2910         JMP NEXT.PATHNAME.CHAR
                2920 *-------------------------------
                2930 SET.NEW.PATHNAME
1147- A9 A0     2940         LDA #" "
1149- 20 ED FD  2950         JSR COUT
114C- 8E 80 02  2960         STX BUF
114F-           2970         >MLI C4,FILE.INFO.PARM
114F- 20 00 BF  0000>         JSR MLI
1152- C4 D1 12  0000>         .DA #$C4,FILE.INFO.PARM
1155- 90 03     2980         BCC .1         ...NO ERRORS
1157- 4C F6 11  2990         JMP PROCESS.ERROR
                3000 *-------------------------------
115A- AD D5 12  3010 .1      LDA FILTYP     FILE.INFO.PARM+4
115D- C9 FF     3020         CMP #$FF
115F- F0 05     3030         BEQ .2         "SYS" FILE
1161- A9 01     3040         LDA #1
1163- 4C F6 11  3050         JMP PROCESS.ERROR
                3060 *-------------------------------
1166- A9 00     3070 .2      LDA #0
1168- 8D EA 12  3080         STA CL.REF     CLOSE.PARM+1, REF NO.
116B-           3090         >MLI CC,CLOSE.PARM
116B- 20 00 BF  0000>         JSR MLI
116E- CC E9 12  0000>         .DA #$CC,CLOSE.PARM
1171- 90 03     3100         BCC .3         ...NO ERROR
1173- 4C F6 11  3110         JMP PROCESS.ERROR
                3120 *-------------------------------
1176- AD D4 12  3130 .3      LDA ACBITS     FILE.INFO.PARM+3
1179- 29 01     3140         AND #1
117B- D0 05     3150         BNE .4         ...OKAY TO READ IT
117D- A9 27     3160         LDA #$27
117F- 4C F6 11  3170         JMP PROCESS.ERROR
                3180 *-------------------------------
1182-           3190 .4      >MLI C8,OPEN.PARM
1182- 20 00 BF  0000>         JSR MLI
1185- C8 E3 12  0000>         .DA #$C8,OPEN.PARM
1188- 90 03     3200         BCC .5         ...NO ERRORS
118A- 4C F6 11  3210         JMP PROCESS.ERROR
                3220 *-------------------------------
118D- AD E8 12  3230 .5      LDA OP.REF     OPEN.PARM+5, REF NO.
1190- 8D EC 12  3240         STA RD.REF     READ.PARM+1, REF NO.
1193- 8D F4 12  3250         STA EF.REF     EOF.PARM+1, REF NO.
1196-           3260         >MLI D1,EOF.PARM
1196- 20 00 BF  0000>         JSR MLI
1199- D1 F3 12  0000>         .DA #$D1,EOF.PARM
119C- 90 03     3270         BCC .6         ...NO ERRORS
119E- 4C F6 11  3280         JMP PROCESS.ERROR
                3290 *-------------------------------
11A1- AD F7 12  3300 .6      LDA FIL.SZ+2     EOF.PARM+4
11A4- F0 05     3310         BEQ .7         ...NOT TOO LONG
11A6- A9 27     3320         LDA #$27
11A8- 4C F6 11  3330         JMP PROCESS.ERROR
                3340 *-------------------------------
11AB- AD F5 12  3350 .7      LDA FIL.SZ       EOF.PARM+2
11AE- 8D EF 12  3360         STA READ.PARM+4
11B1- AD F6 12  3370         LDA FIL.SZ+1     EOF.PARM+3
11B4- 8D F0 12  3380         STA READ.PARM+5
11B7-           3390         >MLI CA,READ.PARM
11B7- 20 00 BF  0000>         JSR MLI
11BA- CA EB 12  0000>         .DA #$CA,READ.PARM
11BD- 08        3400         PHP
```

Viewmaster 80,™ the sharpest 80 column card of them all.



	BUILT-IN SOFTSWITCH	SHIFT KEY SUPPORT	LOW POWER DESIGN	80 COLUMN HOME	7 × 9 DOT MATRIX	LIGHT PEN INPUT TS	80 COLUMN OVERSCORE	INVERSE CHARACTERS	
VIEWMASTER 80	✓	✓	✓	✓	✓	✓	✓	✓	✓
SUPRTERM		✓					✓	✓	
WIZARD 80					✓		✓	✓	
VISION 80	✓	✓			✓				
OMNIVISION		✓					✓	✓	
VIEWMAX 80	✓	✓			✓			✓	
SMARTERM	✓	✓				✓	✓		
VIDEOTERM		✓	✓		✓	✓		✓	

Now, get great resolution and total software compatibility for your Apple II® or II+®.

One look at the chart will give you some of the reasons there's only one smart choice in 80 column cards for your Apple. But the real secret to Viewmaster 80's success is something even better: Total compatibility.

The Viewmaster 80 works with all 80 column applications, including DOS 3.3, PRO-DOS, CP/M, Pascal, WordStar, Format II, Easywriter, Applewriter II, Supertext 80, Zardax, Apple PI, Letter Perfect, dBASE II, Visicalc, Multiplan, and *hundreds* of others.

And the Viewmaster 80 delivers a super sharp, state-of-the-art display with a 7 × 9 character matrix for clear, easily readable characters. Here are just a few of the powerful features the Viewmaster 80 delivers for a great price ($139):

• 80 Characters by 24 lines • Fully compatible with all Apple languages and software • Super sharp 7 × 9 character matrix with true descenders • Highest compatibility with existing 80 column software • Power and input connector for light pen • Very low power consumption • High speed (18 MHZ) scroll rate • Upper and lower case characters with true descenders, both inverse and normal; all on-screen editing functions are supported • User-definable cursor shape • Compatible with Apple II, II+ and IIe • Five year warranty

Call to order today, 9 a.m. to 11 p.m. seven days, or send check or money order to Applied Engineering. MasterCard, VISA and C.O.D. welcome. Texas residents add 5⅛% sales tax. Add $10.00 outside U.S.A.

Æ Applied Engineering
P.O. Box 798, Carrollton, TX 75006
(214) 241-6060

```
11BE-             3410          >MLI CC,CLOSE.PARM
11BE- 20 00 BF    0000>          JSR MLI
11C1- CC E9 12    0000>          .DA #$CC,CLOSE.PARM
11C4- 90 04       3420          BCC .9
11C6- 28          3430          PLP
11C7- 4C F6 11    3440 .8       JMP PROCESS.ERROR
                  3450 *--------------------------------
11CA- 28          3460 .9       PLP
11CB- B0 FA       3470          BCS .8
11CD- 4C 00 20    3480          JMP SYSTEM
                  3490 *--------------------------------
                  3500 BACKSPACE.IN.PATHNAME
11D0- A5 24       3510          LDA CH          UNLESS ALREADY AT BEGINNING
11D2- F0 0F       3520          BEQ .1          ...WE WERE
11D4- CA          3530          DEX
11D5- A9 A0       3540          LDA #" "
11D7- 20 ED FD    3550          JSR COUT
11DA- C6 24       3560          DEC CH
11DC- C6 24       3570          DEC CH
11DE- 20 ED FD    3580          JSR COUT
11E1- C6 24       3590          DEC CH
11E3- 4C EA 10    3600 .1       JMP NEXT.PATHNAME.CHAR
                  3610 *--------------------------------
                  3620 PRINT.MESSAGE
11E6- A2 00       3630          LDX #0
11E8- BD E8 11    3640 MSG.LP   LDA MSG.LP,X
11E9-             3650 MSG.ADDR   .EQ *-2
11EB- F0 08       3660          BEQ .1
11ED- 09 80       3670          ORA #$80
11EF- 20 ED FD    3680          JSR COUT
11F2- E8          3690          INX
11F3- D0 F3       3700          BNE MSG.LP
11F5- 60          3710 .1       RTS
                  3720 *--------------------------------
                  3730 PROCESS.ERROR
11F6- 85 DE       3740          STA ERRCOD
11F8- A9 0C       3750          LDA #12         VTAB 13
11FA- 85 25       3760          STA CV
11FC- 20 8E FD    3770          JSR CROUT       MAKE IT 14
11FF- A5 DE       3780          LDA ERRCOD
1201- C9 01       3790          CMP #1
1203- D0 0C       3800          BNE .1
1205- A9 8C       3810          LDA #ERQT.1
1207- 8D E9 11    3820          STA MSG.ADDR
120A- A9 12       3830          LDA /ERQT.1
120C- 8D EA 11    3840          STA MSG.ADDR+1
120F- D0 26       3850          BNE .3
1211- C9 40       3860 .1       CMP #$40
1213- F0 18       3870          BEQ .2
1215- C9 44       3880          CMP #$44
1217- F0 14       3890          BEQ .2
1219- C9 45       3900          CMP #$45
121B- F0 10       3910          BEQ .2
121D- C9 46       3920          CMP #$46
121F- F0 0C       3930          BEQ .2
1221- A9 A3       3940          LDA #ERQT.2
1223- 8D E9 11    3950          STA MSG.ADDR
1226- A9 12       3960          LDA /ERQT.2
1228- 8D EA 11    3970          STA MSG.ADDR+1
122B- D0 0A       3980          BNE .3          ...ALWAYS
122D- A9 BA       3990 .2       LDA #ERQT.3
122F- 8D E9 11    4000          STA MSG.ADDR
1232- A9 12       4010          LDA /ERQT.3
1234- 8D EA 11    4020          STA MSG.ADDR+1
1237- 20 E6 11    4030 .3       JSR PRINT.MESSAGE
123A- A9 00       4040          LDA #0          VTAB 1
123C- 85 25       4050          STA CV
123E- 4C D1 10    4060          JMP START.PATHNAME.OVER
                  4070 *--------------------------------
1241- C5 CE D4
1244- C5 D2 A0
1247- D0 D2 C5
124A- C6 C9 D8
124D- A0 A8 D0
1250- D2 C5 D3
1253- D3 A0 A2
1256- D2 C5 D4
1259- D5 D2 CE
```

```
125C- A2 A0 D4
125F- CF A0 C1
1262- C3 C3 C5
1265- D0 D4 A9 4080 Q.PRFX .AS -/ENTER PREFIX (PRESS "RETURN" TO ACCEPT)/
1268- 00       4090        .HS 00
1269- C5 CE D4
126C- C5 D2 A0
126F- D0 C1 D4
1272- C8 CE C1
1275- CD C5 A0
1278- CF C6 A0
127B- CE C5 D8
127E- D4 A0 C1
1281- D0 D0 CC
1284- C9 C3 C1
1287- D4 C9 CF
128A- CE       4100 Q.PATH .AS -/ENTER PATHNAME OF NEXT APPLICATION/
128B- 00       4110        .HS 00
128C- 87       4120 ERQT.1 .HS 87
128D- CE CF D4
1290- A0 C1 A0
1293- D4 D9 D0
1296- C5 A0 A2
1299- D3 D9 D3
129C- A2 A0 C6
129F- C9 CC C5 4130        .AS -/NOT A TYPE "SYS" FILE/
12A2- 00       4140        .HS 00
12A3- 87       4150 ERQT.2 .HS 87
12A4- C9 AF CF
12A7- A0 C5 D2
12AA- D2 CF D2
12AD- A0 A0 A0
12B0- A0 A0 A0
12B3- A0 A0 A0
12B6- A0 A0 A0 4160        .AS -"I/O ERROR             "
12B9- 00       4170        .HS 00
12BA- 87       4180 ERQT.3 .HS 87
12BB- C6 C9 CC
12BE- C5 AF D0
12C1- C1 D4 C8
12C4- A0 CE CF
12C7- D4 A0 C6
12CA- CF D5 CE
12CD- C4 A0 A0 4190        .AS -"FILE/PATH NOT FOUND  "
12D0- 00       4200        .HS 00
               4210 *-------------------------------
               4220 FILE.INFO.PARM
12D1- 0A       4230        .DA #10
12D2- 80 02    4240        .DA BUF
12D4- 00       4250 ACBITS .HS 00
12D5- 00       4260 FILTYP .HS 00
12D6-          4270        .BS 13
               4280 *-------------------------------
               4290 OPEN.PARM
12E3- 03       4300        .DA #3
12E4- 80 02    4310        .DA BUF
12E6- 00 18    4320        .DA $1800    BUFFER ADDR
12E8-          4330 OP.REF .BS 1        REF NO.
               4340 *-------------------------------
               4350 CLOSE.PARM
12E9- 01       4360        .DA #1
12EA-          4370 CL.REF .BS 1        REF NO.
               4380 *-------------------------------
               4390 READ.PARM
12EB- 04       4400        .DA #4
12EC-          4410 RD.REF .BS 1        REF NO.
12ED- 00 20    4420        .DA $2000    BUFFER ADDR
12EF-          4430        .BS 2        # BYTES TO READ
12F1-          4440        .BS 2        # ACTUALLY READ
               4450 *-------------------------------
               4460 EOF.PARM
12F3- 02       4470        .DA #2
12F4-          4480 EF.REF .BS 1        REF NO.
12F5-          4490 FIL.SZ .BS 3        EOF POSITION
               4500 *-------------------------------
               4510 PREFIX.PARM
12F8- 01       4520        .DA #1
12F9- 80 02    4530        .DA BUF
               4540 *-------------------------------
```

Two Ways to Merge Fields in a Byte.....Bob Sander-Cederlof

One of the advantages of assembly language is that data can be manipulated easily at the bit and byte level. This leads to efficiencies in both speed and memory usage which cannot be matched with most higher-level languages.

We can pack more than one data item into the same byte. For example, I may use the first three bits of a byte to indicate which of eight colors to use, and the other five bits to indicated position on a 32-pixel line. There are endless examples. Since we need to be able to store into and retrieve from bit-fields within bytes, all of the microprocessors include opcodes which make it possible.

To merge two values together which already are "clean", we simply use the ORA opcode. For example, if I have data for field A in VAL.A as xxx00000 and data for field B in VAL.B as 000xxxxx, I merge them like this:

```
        LDA VAL.A
        ORA VAL.B
```

By "clean" I mean that all the bits in VAL.A and VAL.B which are not part of the field values are already zero. If they are not, then we must first strip out those bits with the AND opcode:

```
        LDA VAL.A
        AND #$E0
        STA TEMP
        LDA VAL.B
        AND #$1F
        ORA TEMP
```

There is another way, which is shorter and faster and does not need TEMP. However, it is harder to figure out why it works.

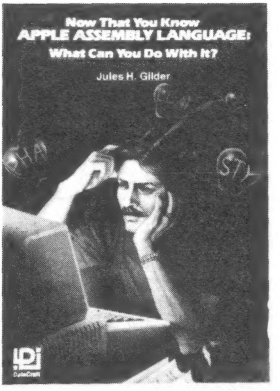
```
        LDA VAL.A
        EOR VAL B
        AND #$1F
        EOR VAL.A
```

Can you explain it? I was so unsure of myself when I first ran into this technique that I devised a test program. My test tries all 256 values of VAL.A and VAL.B, with all possible contiguous fields from 1 bit for VAL.A to 7 bits for VAL.A. Probably overkill, but it runs in a few seconds.

My program prints out the two field masks for each of the seven field sizes, so that I can tell it is running. If the two methods for merging get the same results, that is the only output. If they do not, indicating that one method or the other does not work, I print out more data.

While I was writing the program I tried several variations, such as printing all the results whether they agreed or not. In order to be able to look at that volume of output

BECOME AN ASSEMBLY LANGUAGE PROGRAMMING WHIZ



You've spent a lot of time learning Apple assembly language and finally know the difference between BEQ and BCS. Now it's time to put your new-found knowledge to work. Time to throw away your Applesoft programming manual and write programs that make your Apple work like a super-charged, super-fast computer. Time to graduate from the Applesoft BASIC used by beginners, to the 6502 assembly language used by professionals.

To help make this transition, you need an experienced programmer to guide you. You need to develop a library of subroutines that make programming in assembly language as easy as programming in BASIC. You need to learn all the tricks that take experienced assembly language programmers years to acquire. Most important of all, you need the book, *"Now that You Know Apple Assembly Language: What Can You Do With It?"* because it contains all this information *and more.*

It shows you how, step-by-step

"Now That You Know Apple Assembly Language: What Can You Do With It?" will take you step-by-step through the assembly language programming experience. You'll delve into the mysteries of the 6502 stack and learn how to use it to increase the power and versatility of your programs. You'll also learn how to use the Apple's built-in routines to minimize the amount of coding you must do.

Control the output and the input

Frequently it's desirable to gain total control of the computer's output. This book shows you how to *steal control away from the Apple's normal output routines and redirect it to your own program.* Thus if you wanted, you could see the normally invisible control characters, display text on your screen as black on white instead of the normal white on black, format text sent to a printer into pages and much more.

Expand the power of your Apple by *stealing control away from the normal input routines.* Do things like adding a screen print capability, or *convert part of the normal keyboard into a numeric keypad.* It's even possible to *produce self-modifying programs* by EXECing in commands from RAM instead of from the disk drive. Think about the possibilities that offers for protecting your programs. When you want to go back to Applesoft programming, *you'll be able to do it faster with the aid of Applesoft Shorthand,* an assembly language program that types in one or more Applesoft commands at the press of a key, or use another program in the book to *automatically count the number of lines in your Applesoft program.*

With this book you'll also learn about *generating tones and how to figure out the frequency, producing sound effects, teaching your Apple to send Morse code, restoring accidentally erased Applesoft programs, adding new commands to Applesoft and running two Applesoft programs in memory together,* to name a few.

Everything is explained

Unlike other books that merely consist of a collection of programs, this one explains what's happening, where and why. You get detailed descriptions of how the programs work and detailed program listings with virtually every line of code explained. Nothing is left to chance or misinterpretation.

Order now, get 2 FREE gifts

The book costs only $19.95 plus $2 for shipping and handling. Order now and you'll also get a *FREE Programmer's Number Conversion System* that makes it easy to convert between binary, hexadecimal and decimal numbers. No calculators are required. You'll convert numbers almost instantly and wonder how you ever got along without it.

As an extra bonus for prompt ordering, you'll receive a *FREE coupon worth $5 off* the price of a disk with all the assembled programs on it or a disk that contains the source code. These disks normally sell for $15 each. We're offering these FREE gifts for a limited time only, so hurry! *Order today!*

Money-back guarantee*

We're so confident that you'll find this book invaluable and want it in your library, that we're offering a 10-day, no-questions-asked, money-back guarantee. Order the book. Read it and try the programs for ten days. At the end of ten days if you don't think it's worth every penny you paid for it, just send it back in resalable condition and we'll refund your money immediately, no questions asked.

Redlig Systems, Inc., Dept. A 978
2068—79th St., Brooklyn, NY 11214

Please rush me _____ copies of **"Now That You Know Apple Assembly Language: What Can You Do With It?"** at $19.95 each plus $2 shipping and handling. I understand that if I am not delighted with the book I may return it within 10 days for a prompt and courteous refund. In any case, the Programmer's Number Conversion System and $5 coupon are mine to keep.

☐ Enclosed is my check for $ _____

Please charge my credit card:
☐ American Express ☐ MasterCard ☐ Visa

Card No. _____ Exp. _____

Signature _____

Name _____

Address _____

City _____ State _____ Zip _____

NOTE: Shipping and handling fees are not refundable.

reasonably, I added a PAUSE subroutine which enabled me to stop the output by tapping any key, restart it the same way, and abort by tapping the RETURN key.

The code for the first merging method is in lines 1310-1380; that for the second at lines 1400-1450.

The test was conclusive. I tried every possible combination, and both methods always give the same results. Looking back, I can see that the whole test was unnecessary; the second method will OBVIOUSLY produce the same results. Now I see it. Do you?

```
                    1000 *SAVE MERGE FIELDS IN A BYTE
                    1010 *-------------------------------
FD8E-               1020 CROUT  .EQ $FD8E
FDDA-               1030 PRBYTE .EQ $FDDA
FDED-               1040 COUT   .EQ $FDED
                    1050 *-------------------------------
00-                 1060 FIELD.A    .EQ $00
01-                 1070 FIELD.B    .EQ $01
02-                 1080 VAL.A      .EQ $02
03-                 1090 VAL.B      .EQ $03
04-                 1100 MERGE.1    .EQ $04
05-                 1110 MERGE.2    .EQ $05
                    1120 *-------------------------------
                    1130 T
                    1140 *---FOR FIELD= 80,7F TO 7F,80----
0800- A9 7F         1150        LDA #$7F     DEFINE FIELDS AS 1,7
0802- 85 01         1160        STA FIELD.B
0804- A9 80         1170        LDA #$80
0806- 85 00         1180        STA FIELD.A
                    1190 *---FOR A=0 TO MAX VAL-----------
0808- A9 00         1200 .1     LDA #0
080A- 85 02         1210        STA VAL.A
080C- 20 8E FD      1220        JSR CROUT
080F- A5 00         1230        LDA FIELD.A
0811- 20 68 08      1240        JSR PRBYTESP
0814- A5 01         1250        LDA FIELD.B
0816- 20 DA FD      1260        JSR PRBYTE
                    1270 *---FOR B=0 TO MAX VAL-----------
0819- A9 00         1280 .2     LDA #0
081B- 85 03         1290        STA VAL.B
                    1300
                    1310 *---MERGE FIRST METHOD-----------
081D- A5 02         1320 .3     LDA VAL.A
081F- 25 00         1330        AND FIELD.A
0821- 85 04         1340        STA MERGE.1
0823- A5 03         1350        LDA VAL.B
0825- 25 01         1360        AND FIELD.B
0827- 05 04         1370        ORA MERGE.1
0829- 85 04         1380        STA MERGE.1
                    1390
                    1400 *---MERGE SECOND METHOD----------
082B- A5 02         1410        LDA VAL.A
082D- 45 03         1420        EOR VAL.B
082F- 25 01         1430        AND FIELD.B
0831- 45 02         1440        EOR VAL.A
0833- 85 05         1450        STA MERGE.2
                    1460
                    1470 *---PRINT RESULTS, IF NOT EQUAL--
0835- C5 04         1480        CMP MERGE.1
0837- F0 1F         1490        BEQ .4
0839- 20 8E FD      1500        JSR CROUT
083C- A5 00         1510        LDA FIELD.A
083E- 20 68 08      1520        JSR PRBYTESP
0841- A5 02         1530        LDA VAL.A
0843- 20 68 08      1540        JSR PRBYTESP
0846- A5 03         1550        LDA VAL.B
0848- 20 68 08      1560        JSR PRBYTESP
084B- A5 04         1570        LDA MERGE.1
084D- 20 68 08      1580        JSR PRBYTESP
0850- A5 05         1590        LDA MERGE.2
0852- 20 DA FD      1600        JSR PRBYTE
0855- 20 70 08      1610        JSR PAUSE
```

```
                        1620 *---NEXT B--------------------
0858- E6 03             1630 .4     INC VAL.B
085A- D0 C1             1640        BNE .3
                        1650 *---NEXT A--------------------
085C- E6 02             1660        INC VAL.A
085E- D0 B9             1670        BNE .2
                        1680 *---NEXT FIELD----------------
0860- 38                1690        SEC
0861- 66 00             1700        ROR FIELD.A
0863- 46 01             1710        LSR FIELD.B
0865- D0 A1             1720        BNE .1          CONTINUE
0867- 60                1730        RTS             FINISHED
                        1740 *-------------------------------
                        1750 PRBYTESP
0868- 20 DA FD          1760        JSR PRBYTE
086B- A9 A0             1770        LDA #$A0
086D- 4C ED FD          1780        JMP COUT
                        1790 *-------------------------------
0870- AD 00 C0          1800 PAUSE  LDA $C000
0873- 10 16             1810        BPL .3
0875- 8D 10 C0          1820        STA $C010
0878- C9 8D             1830        CMP #$8D
087A- D0 03             1840        BNE .2
087C- 68                1850 .1     PLA
087D- 68                1860        PLA
087E- 60                1870        RTS
087F- AD 00 C0          1880 .2     LDA $C000
0882- 10 FB             1890        BPL .2
0884- 8D 10 C0          1900        STA $C010
0887- C9 8D             1910        CMP #$8D
0889- F0 F1             1920        BEQ .1
088B- 60                1930 .3     RTS
                        1940 *-------------------------------
```


Now you can monitor and control the world (or at least your part of it) with a little help from

APPLIED ENGINEERING

12 BIT, 16 CHANNEL, PROGRAMMABLE GAIN A/D

- All new 1984 design incorporates the latest in state-of-art I.C. technologies.
- Complete 12 bit A/D converter, with an accuracy of 0.02%!
- 16 single ended channels (single ended means that your signals are measured against the Apple's GND.) or 8 differential channels. Most all the signals you will measure are single ended.
- 9 software programmable full scale ranges, any of the 16 channels can have any range at any time. Under program control, you can select any of the following ranges: ±10 volts, ±5V, ±2.5V, ±1.0V, ±500MV, ±250MV, ±100MV, ±50MV, or ±25MV.
- Very fast conversion (25 micro seconds).
- Analog input resistance greater than 1,000,000 ohms.
- Laser-trimmed scaling resistors.
- Low power consumption through the use of CMOS devices.
- The user connector has +12 and -12 volts on it so you can power your sensors.
- Only elementary programming is required to use the A/D.
- The entire system is on one standard size plug in card that fits neatly inside the Apple.
- System includes sample programs on disk.

PRICE $319

A few applications may include the monitoring of ● flow ● temperature ● humidity ● wind speed ● wind direction ● light intensity ● pressure ● RPM ● soil moisture and many more.

A/D & D/A

A/D & D/A Features:

- Single PC card
- 8 channels A/D
- 8 channels D/A
- Superfast conversion time
- Very easy programming
- Many analog ranges
- Manual contains sample applications

A/D SPECIFICATIONS

- 0.3% accuracy
- On-board memory
- Fast conversion (.078 MS per channel)
- A/D process totally transparent to Apple (looks like memory)
- User programmable input ranges are 0 to 10 volts, 0 to 5, - 5 to + 5, 2.5 to + 2.5, - 5 to 0, - 10 to 0.

The A/D process takes place on a continuous, channel sequencing basis. Data is automatically transferred to its proper location in the on-board RAM. No A/D converter could be easier to use.

D/A SPECIFICATIONS

- 0.3% accuracy
- On-board memory
- On-board output buffer amps can drive 5 MA
- D/A process is totally transparent to the Apple (just poke the data)
- Fast conversion (.001 MS per channel)
- User programmable output ranges are 0 to 5 volts and 0 to 10 volts

The D/A section contains 8 digital to analog converters, with output buffer amplifiers and all interface logic on a single card. On-card latches are provided for each of the eight D/A converters. No D/A converter could be easier to use. The on-board amplifiers are laser-trimmed during manufacture, thereby eliminating any requirement for off-set nulling.

PRICE $199

SIGNAL CONDITIONER

Our 8 channel signal conditioner is designed for use with both our A/D converters. This board incorporates 8 F.E.T. op-amps, which allow almost any gain or offset. For example, an input signal that varies from 2.00 to 2.15 volts or a signal that varies from 0 to 50 mV can easily be converted to 0-10V output for the A/D.

The signal conditioner's outputs are on a high quality 16 pin gold I.C. socket that matches the one on the A/D's so a simple ribbon cable connects the two. The signal conditioner can be powered by your Apple or from an external supply.

FEATURES

- 4.5" square for standard card cage and 4 mounting holes for standard mounting. The signal conditioner does not plug into the Apple, it can be located up to ½ mile away from the A/D.
- 22 pin .156 spacing edge card input connector (extra connectors are easily available i.e. Radio Shack).
- Large bread board area.
- Full detailed schematic included.

PRICE $79

I/O 32

- Provides 4, 8-Bit programmable I/O Ports
- Any of the 4 ports can be programmed as an input or an output port
- All I/O lines are TTL (0-5 volt) compatible
- Your inputs can be anything from high speed logic to simple switches
- Programming is made very easy by powerful on-board firmware
- The I/O 32 is your best choice for any control application

The I/O manual includes many programs for inputs and outputs.

Some applications include:
Burglar alarm, direction sensing, use with relays to turn on lights, sound buzzers, start motors, control tape recorders and printers, use with digital joystick.

PRICE $89

Please see our other full page ad in this magazine for information on Applied Engineering's Timemaster Clock Card and other products for the Apple.
Our boards are far superior to most of the consumer electronics made today. All I.C.'s are in high quality sockets with mil-spec. components used throughout. P.C. boards are glass-epoxy with gold contacts. Made in America to be the best in the world. All products compatible with Apple II and //e.

Applied Engineering's products are fully tested with complete documentation and available for immediate delivery. All products are guaranteed with a no hassle three year warranty

Texas Residents Add 5% Sales Tax
Add $10.00 If Outside U.S.A.

Send Check or Money Order to:
APPLIED ENGINEERING
P.O. Box 798
Carrollton, TX 75006

Call (214) 492-2027
7 a.m. to 11 p.m. 7 days a week
MasterCard, Visa & C.O.D. Welcome
No extra charge for credit cards

Comments on O'Ryan's 65C02 Mod for Apple II.....Jim Sather

William O'Ryan's method (October 1985 AAL) of modifying old Apples to accept 65C02s looks like a very reliable fix. I notice no negative consequences in RAM or video timing. I do however recommend switching to 150 nanosecond motherboard RAM.

Apple motherboard RAM read access is CAS' limited, meaning TCAC (delay from CAS' falling to read data valid) is the critical RAM chip specification. In an Apple with O'Ryan's fix, RAM chips have 140 nsec minus 74LS139 pin 1 to pins 4,5,6 high/low propagation delay to get RAM read data valid after CAS' falls at the RAM chips. This means TCAC needs to be 119 nsec or less with a typical LS139. TCAC specifications are 100 nsec for 150 nsec RAM and 135 nsec for 200 nsec RAM, so 150 nsec or faster chips should be installed to be within RAM chip specifications with O'Ryan's fix.

A given Apple II may work with O'Ryan's fix and 200 nsec RAM chips, but operation may not be reliable over a wide range of room temperatures. Again I say, O'Ryan's fix calls for 150 nsec RAM chips. To operate with slower chips is asking for trouble.

Incidentally, 16K RAM chips don't cost as much as they used to. The cheapest 150 nsec 16K RAM chips I can find in my current mail order catalogs are 45 cents apiece at Jameco Electronics, 1355 Shoreway Rd., Belmont, CA 94002. [Slower ones were $65.00 apiece in 1978!]

As an alternative to replacing slow motherboard RAM chips, one can replace the 74LS139 at F2 with a 74S139. This changes the TCAC requirement with O'Ryan's fix to 133 nsec for a typical S139, and to 130 nsec for a worst case S139. These are barely less than the 135 nsec specification of 200 nsec RAM, so operation with 200 nsec RAM is probably reliable.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION. P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $18 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $14 postage for other countries. Back issues are available for $1.80 each (other countries add $1 per back issue for postage).

All material herein is copyrighted by S-C SOFTWARE CORPORATION, all rights reserved. (Apple is a registered trademark of Apple Computer, Inc.)